Escuela Politécnica Superior

Sistema seguro para la gestión de datos sensibles de forma colaborativa

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
Carlos Zamora Sanz

Tutor/es:
Rafael Ignacio Álvarez Sánchez

Junio 2021

Universitat d'Alacant
Universidad de Alicante

Mi más sincero agradecimiento:

A mi tutor, Rafa,
por su constante
orientación
y especial dedicación,
que han contribuido,
sin duda, a
enriquecer la llegada
a puerto de
este trabajo.

A mi familia
y amigos
por animarme
y ayudarme
en los momentos
más duros.

Índice

Índice de figuras

1.Introducción

En la gestión segura de datos sensibles, es cada vez más importante la colaboración entre distintas entidades para el intercambio de información, a la vez que se garantiza la privacidad de los usuarios. En este ámbito, cabe destacar tecnologías como Blockchain [1] y las autoridades de certificación o infraestructuras de clave pública o PKI [2].

En este trabajo de fin de máster se ha desarrollado un proyecto que amplía el sistema creado con anterioridad en el trabajo de fin de grado. El sistema implementado permite ofrecer un servicio de gestión segura de datos sensibles sanitarios en el que se han enfatizado las tareas relacionadas con la colaboración y el intercambio de información, asegurando la privacidad, integridad y autenticidad de la información y atendiendo a otros aspectos como la integración de servicios, la escalabilidad o la ingeniería del software.

Con el uso generalizado de las nuevas tecnologías, la sensibilidad hacia la privacidad de los usuarios se ha convertido en un tema prioritario y la gestión segura de la información generada en un problema muy relevante, considerándose en muchas situaciones un problema crítico.

Un campo especialmente sensible es el de los datos sanitarios en el que se conocen casos recientes de robo y tráfico ilegal de datos a lo largo de todo el mundo. Tal es el caso reconocido por parte de la administración sanitaria noruega en 2018 del acceso no autorizado a datos sanitarios de casi tres millones de ciudadanos como consecuencia de un fallo de seguridad en su web [3] o de la exposición de datos sanitarios de dos millones de mujeres chinas en 2019 al detectarse un agujero de seguridad en la base de datos Breed Ready [4] o de las 96 fugas de datos en el sector sanitario estadounidense ocurridas durante el primer cuatrimestre de 2019 que afectaron a, aproximadamente, tres millones y medio de usuarios.

El volumen de datos personales expuestos en el mundo cada año supera al anterior, considerándose, por ejemplo, en un informe de Cesicat [5] que el número de datos personales expuestos en 2019, unos 2700 millones, duplicó a los expuestos en 2018 que fueron unos 1300 millones; estimándose que un 30% de los mismos son datos personales de carácter sanitario.

Uno de los principales móviles que explican este tipo de ataques es el económico, pero existen, sin duda, muchos más motivos que, en el caso de los datos sanitarios, recorren diversos caminos como la creación de identidades falsas o la sustracción de expedientes con información sensible con el fin de estafar o extorsionar bien a los individuos, que desean proteger su privacidad; bien a las entidades médicas, que precisan recuperar la información que están obligadas a proteger.

Los patrones más repetidos que permiten los ataques descritos con anterioridad consisten en la utilización de claves sencillas o almacenadas en claro lo que facilita, en caso de acceso indebido a las mismas, su utilización en diversas aplicaciones como el almacenamiento de datos sensibles en bases de datos no cifradas, pudiendo relacionar los datos con los usuarios.

La realización de este trabajo ha tenido diversas motivaciones, la primera de la cuales es mi inquietud e interés por los temas relacionados con la seguridad de la información. Otra motivación viene de la experiencia adquirida en la realización de prácticas de empresa, ya que trabajé en proyectos en los que se realizaba tratamiento de datos sanitarios y comprobé que la seguridad no era un valor importante en las aplicaciones involucradas. Sin duda, que quedaran algunas propuestas de ampliación en el proyecto desarrollado en el trabajo de fin de grado, ha definido el desarrollo del presente trabajo que se ha centrado en la compartición con consentimiento de datos sanitarios entre entidades médicas mediante el uso de infraestructuras de clave pública y la mejora del sistema de *logs* con el objetivo principal de garantizar el no repudio.

2.Estado del arte

En los siguientes apartados se describen, someramente, algunos de los sistemas de infraestructura de clave pública más extendidos; centrándose, por no alargar en exceso la sección, en el ámbito territorial más próximo.

2.1 Blockchain

Es bien conocido que se define como Blockchain o cadena de bloques a una estructura de datos cuya información es agrupada en conjuntos de bloques a los que se le adjunta metainformación relativa a un bloque adherido con anterioridad. Resumidamente se puede decir que una Blockchain es una base de datos distribuida y cifrada, que puede ser utilizada en trámites económicos, administrativos, etc. En la actualidad las Blockchain son utilizadas en sistemas en los que se pretende intercambiar información entre varios sujetos eliminando cualquier intermediario, descentralizando toda la gestión. Algunos ejemplos de aplicaciones que utilizan este tipo de sistemas son: bases de datos distribuidas, licitaciones en contrataciones públicas, notarías públicas a la hora de realizar transacciones, uso de criptomonedas como Bitcoin [6], Ethereum [7] o Dogecoin, etc.

De entre todas sus aplicaciones Bitcoin es la más conocida a nivel mundial, ya que es la criptomoneda con una mayor cotización y permite contabilizar y transmitir valor. Bitcoin se basa en una red de nodos descentralizada en la que cada nodo almacena una copia de todas las transacciones realizadas. Las personas que forman parte de los nodos tienen el incentivo de que cada vez que se generan Bitcoins (la unidad monetaria de esta criptomoneda) nuevos son repartidos entre quienes forman parte de esos nodos.

Ethereum es una plataforma global descentralizada de código abierto utilizada para programar contratos inteligentes, es decir, los desarrolladores pueden hacer uso de ella para codificar nuevos tipos de aplicaciones descentralizadas. Su criptomoneda nativa es el Ether (ETH) y, actualmente, es la segunda criptomoneda más cotizada a nivel mundial. Como en otras criptomonedas, Ether hace uso de un libro digital compartido, que es de acceso público y completamente transparente, en el que se anotan todas las transacciones. Dicho libro contable es una cadena de bloques o Blockchain y se construye mediante el proceso de minería de datos.

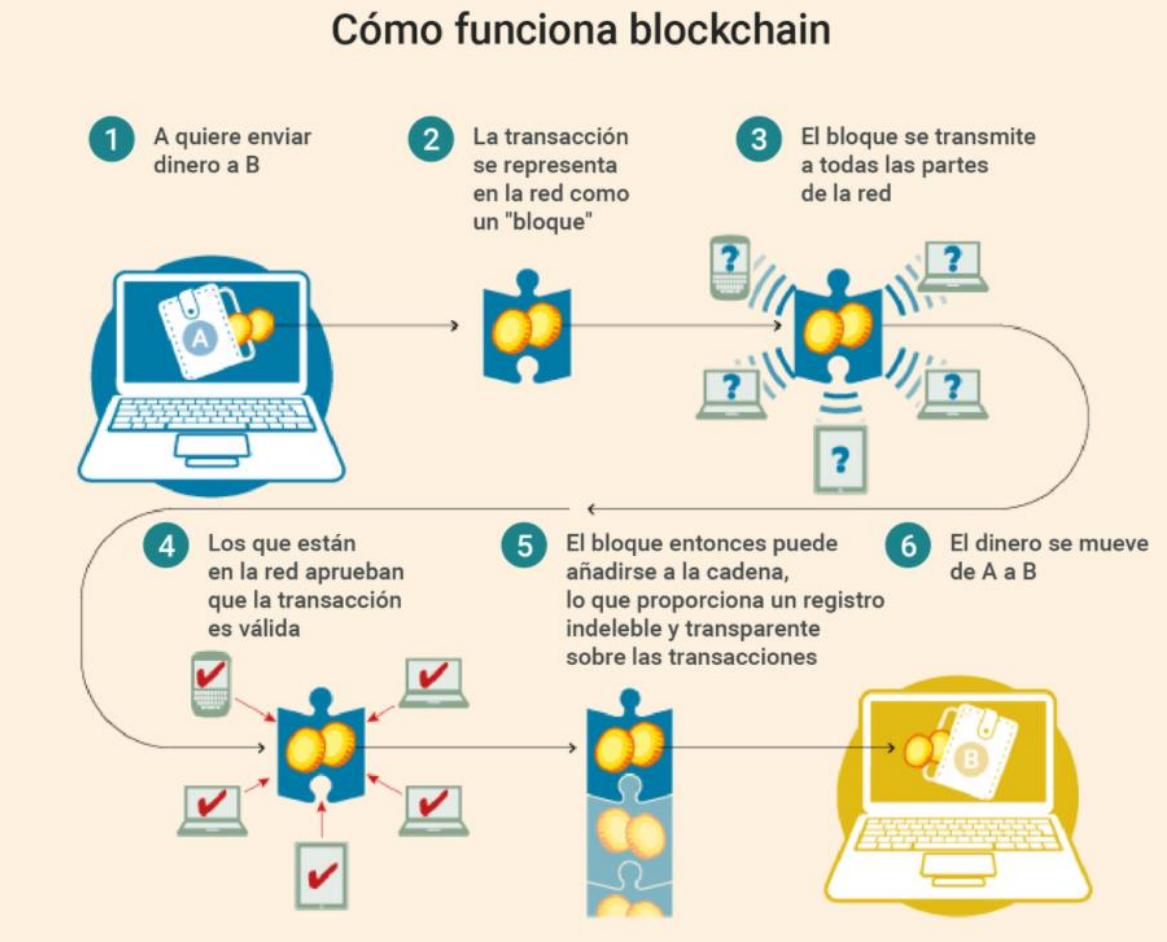

Figura 1.- Esquema del funcionamiento de una Blockchain

2.2 Firma digital

Se puede definir la firma digital [8] como una secuencia de bits añadida a un registro de información, de cualquier índole, que permite garantizar su autenticidad independientemente del proceso de transmisión de datos que se produzca. La firma digital debe cumplir las siguientes propiedades:

- Debe ir ligada indisolublemente al mensaje, es decir, una firma válida para un registro no puede ser válida para otro distinto.
- Ha de ser públicamente verificable. Cualquier entidad debe poder comprobar su autenticidad de forma sencilla.
- Solo puede ser generada por su legítimo propietario.

En la actualidad la firma digital o firma electrónica nos permite realizar una amplia variedad de trámites burocráticos de manera telemática, que anteriormente debían ser realizados de forma presencial.

En el contexto de nuestro país, España, cualquier ciudadano tiene derecho a obtener un certificado digital de persona física a través de la FNMT (Fábrica Nacional de Moneda y

Timbre) de forma gratuita, y puede ser utilizado para firmar documentos privados o para la consulta de información desde plataformas gubernamentales (página web de la Seguridad Social, Clases pasivas, etc.). También existe el DNI electrónico [9], un documento emitido por la Dirección General de la Policía que acredita físicamente la identidad personal del titular que lo posee. Entre otras cosas, permite acreditar electrónicamente de forma inequívoca a su titular y firmar digitalmente documentos electrónicos, otorgándoles validez jurídica de la misma forma que una firma manuscrita. A través de este documento es posible realizar gestiones online de forma segura con las Administraciones Públicas y empresas privadas, así como con otros ciudadanos.

Figura 2.- Interfaz de la aplicación AutoFirma, que puede utilizar el DNI electrónico como método de autenticación

3.Análisis de soluciones de autenticación

Con el auge de las criptomonedas en los últimos años, la tecnología Blockchain ha pasado de ser una solución aplicada en contextos muy puntuales a ser una de las tecnologías más demandadas, llegando incluso a instaurar la creencia de que Blockchain es una tecnología que revolucionará la informática tal y como la comprendemos.

Los grupos que desarrollan soluciones basadas en Blockchain o de PKI (*Publ*ic *Key Infraestructure*) distribuidas afirman que reemplazarán gradualmente a los sistemas PKI tradicionales, resolviendo problemas ligados a estos como:

- La dificultad de gestionar el sistema.
- Los problemas a la hora de escalar el sistema en un entorno de aplicación distribuida.
- Debido a su rol crítico en estos sistemas, las autoridades certificadoras pueden ser un objetivo muy atractivo para el ciberataque.
- Las compañías descuidan la gestión de los certificados en ocasiones.

En contraposición, se argumenta que Blockchain o las PKI distribuidas eliminan los riesgos de los sistemas PKI tradicionales porque:

- Cuentan con una arquitectura abierta, transparente y segura.
- Si se emite un certificado en nombre de un tercero, todos los miembros de la cadena son conscientes de ello.
- Cualquier usuario puede leer el contenido, no es necesario confiar en una Autoridad Certificadora.

Pese a esta argumentación, los sistemas Blockchain o de PKI distribuida no abordan las cuestiones críticas de cualquier implementación PKI.

Por ejemplo, un aspecto negativo de Blockchain es la alta latencia que sufren las transacciones, ya que la rapidez de este tipo de acciones es un aspecto fundamental para el éxito de la gran mayoría de sistemas. Dependiendo del sistema Blockchain se puede alcanzar una velocidad de hasta 500000 transacciones por segundo, cifra que puede ser rápida si se compara con otros sistemas Blockchain pero insuficiente si es comparada con una base de datos de un sistema centralizado. Incluso si los sistemas Blockchain aumentan su velocidad de cómputo en un futuro, es altamente probable que no lleguen a la misma velocidad que puede ofrecer un sistema centralizado, ya que requieren de un trabajo adicional. Por ejemplo, Bitcoin requiere que el usuario con menor capacidad de cómputo realice una prueba de trabajo que asegure la cadena de bloques, forzando a que la velocidad de minería del sistema se ajuste a dicha capacidad.

Una desventaja de los sistemas de PKI distribuida radica en su seguridad, ya que una vez que se introduce información en la cadena de bloques todos pueden verla. Esto puede provocar que algunos aspectos, que son simples en cualquier modelo centralizado, sean mucho más difíciles de implementar. Además, si un minero obtiene la información de la transacción con la que predecir el resultado de un contrato inteligente, puede adelantar esa misma transacción retrasando la del usuario legítimo y colocando una transacción prefabricada en su lugar para explotar su conocimiento previo en la Blockchain.

Otro aspecto negativo de los sistemas Blockchain es la necesidad de almacenar una cadena de bloques que cada vez ocupa más, volviéndose lentas y difíciles de manejar conforme aumentan de tamaño y aumenta el volumen de usuarios que acceden y escriben en ella. Se espera que este problema sea resuelto con los avances en ingeniería y velocidad de procesamiento que vayan sucediendo, pero por el momento es un problema a tener en cuenta.

Además, en la actualidad este tipo de sistemas carecen de una reglamentación legal seria, lo que los convierte en un entorno de riesgo, siendo comunes las estafas o manipulaciones del mercado.

Si bien los sistemas de PKI distribuido presentan serios problemas, existen casos de uso en los que pueden proporcionar buenos resultados, como en la publicación de registros de transparencia de certificados o la distribución de listas de revocación de certificados o CRL.

Se puede concluir que las soluciones distribuidas no proporcionan ventajas significativas frente a soluciones tradicionales en la mayoría de aplicaciones, sino que resuelven problemas que en gran parte son debidos a errores de implementación, falta de automatización o fallas en los sistemas heredados; dichos problemas pueden ser abordados y resueltos manteniendo un sistema centralizado, con una implementación segura y robusta. Por ello, se ha optado por desarrollar un sistema PKI centralizado, en el que existe una Autoridad Certificadora común a todos los nodos del sistema.

4.Planificación

En este apartado se detallan los métodos de desarrollo utilizados durante el proyecto, así como las especificaciones aproximadas y realistas del proceso de implementación del proyecto en el entorno de producción.

4.1 Metodología de desarrollo

Durante el desarrollo del proyecto se ha tratado de seguir, en líneas generales, los principios de la metodología de desarrollo ágil XP (Programación Extrema o Extreme Programming). La metodología que se ha empleado ha sido impartida en asignaturas como GPI (Gestión de Proyectos Informáticos) o AESS (Análisis y Especificación de Sistemas Software), cursadas durante el grado de Ingeniería Informática.

Una de las características de XP es que se pone especial énfasis en promover la retroalimentación entre el equipo de desarrollo y el cliente. Esta característica se ha cumplido a grandes rasgos durante el desarrollo del proyecto, en gran parte debido a que la comunicación entre tutor y alumno ha sido muy frecuente. También cabe destacar que XP es una metodología apropiada para proyectos con requisitos cambiantes o imprecisos.

Al emplear XP, se realizan una sucesión de iteraciones en la que se definen objetivos, concretando unas fechas de entrega que no han de ser cumplidas de forma exacta, pero sirven a modo de orientación. Esto ha ocurrido en el proyecto de forma frecuente, ya que el modo de trabajo ha consistido en reuniones en las que se definían varios objetivos y se especificaba una fecha de cumplimiento de estos. Generalmente las reuniones se han producido con una oscilación de 2 a 4 semanas, en función de la carga de trabajo de tutor y alumno. En la Figura 3 se muestra un diagrama en el que se explica la metodología de desarrollo empleada en el proyecto.

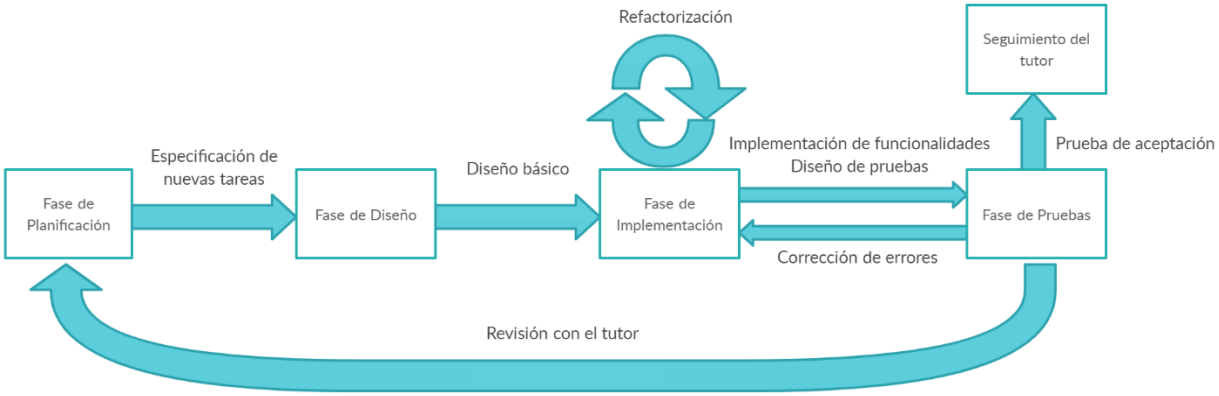

Figura 3.- Diagrama de la metodología de desarrollo empleada en el proyecto.

Debido a las limitaciones de movilidad, aplicadas desde el inicio del curso, se tuvo que sustituir las tutorías presenciales por tutorías en línea, concertándolas mediante correos electrónicos con una frecuencia de entre 2 y 4 semanas. En dichas reuniones se presentaban preguntas con respecto a la implementación del proyecto y se resolvían dudas planteadas por el alumno.

Para gestionar las tareas a realizar, planteadas a lo largo de todo el proyecto, se ha decidido utilizar Trello [10]. Esta plataforma presenta un tablón de trabajo en el que se pueden ir añadiendo columnas con un nombre asignado, que en el caso del proyecto representan los diferentes estados de la tarea (por hacer, en curso, pendiente de evaluación, hecho). En primer lugar, se ha añadido la tarea en cuestión a la lista "Por hacer" y después ha ido pasando entre cada una de listas restantes, hasta llegar a la lista "Hecho". En la figura 3 se ilustra el tablón de Trello utilizado durante la realización del proyecto.

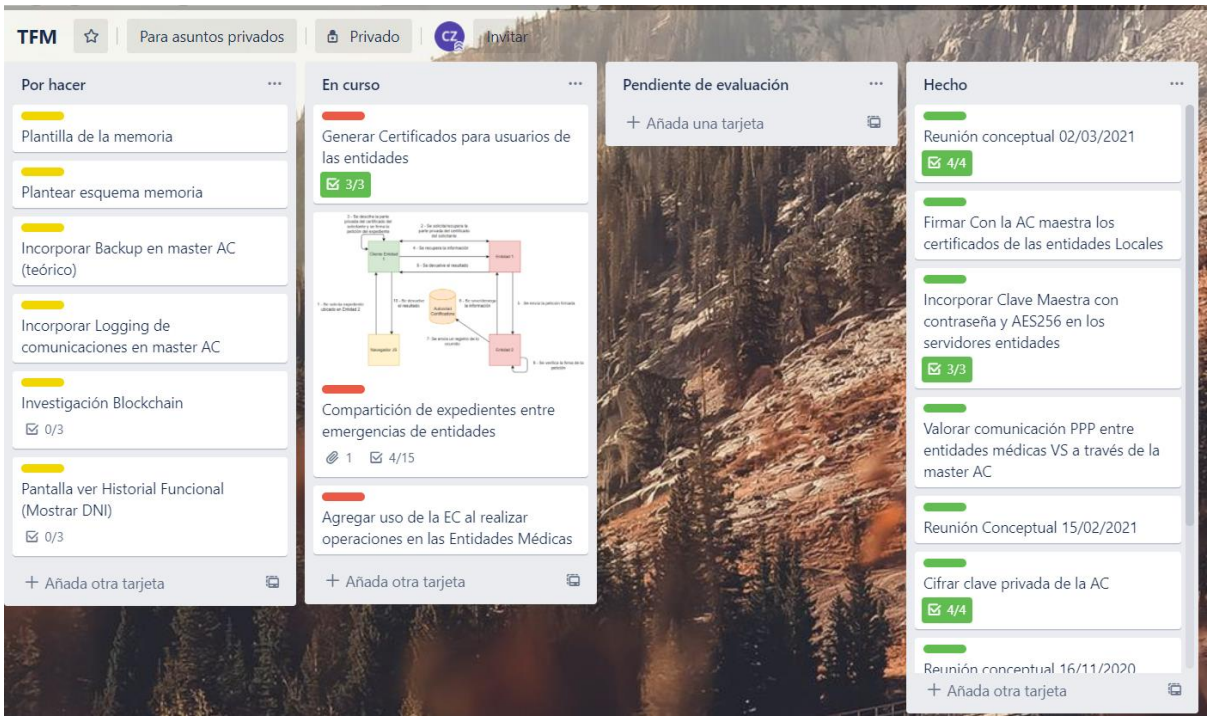

Figura 4.- Panel Trello empleado para organizar las tareas del proyecto.

Además, para gestionar el control de versiones de la implementación del sistema se ha empleado Github [11], que es una herramienta ampliamente conocida, gratuita y que cuenta con una gran comunidad de usuarios. Esta herramienta ha sido utilizada en varias asignaturas durante el trascurso del máster, como Protección de la Información o Desarrollo de Aplicaciones Seguras. La mecánica de trabajo utilizada durante el desarrollo ha consistido en la realización de *commits* tras la implementación de una nueva funcionalidad, lo que permite separar los puntos de restauración de una forma concisa, aunque en ningún momento se ha tenido que hacer uso de ellos. En la figura 4 se ilustra la página web en la que se ubica el repositorio empleado durante el desarrollo del sistema.

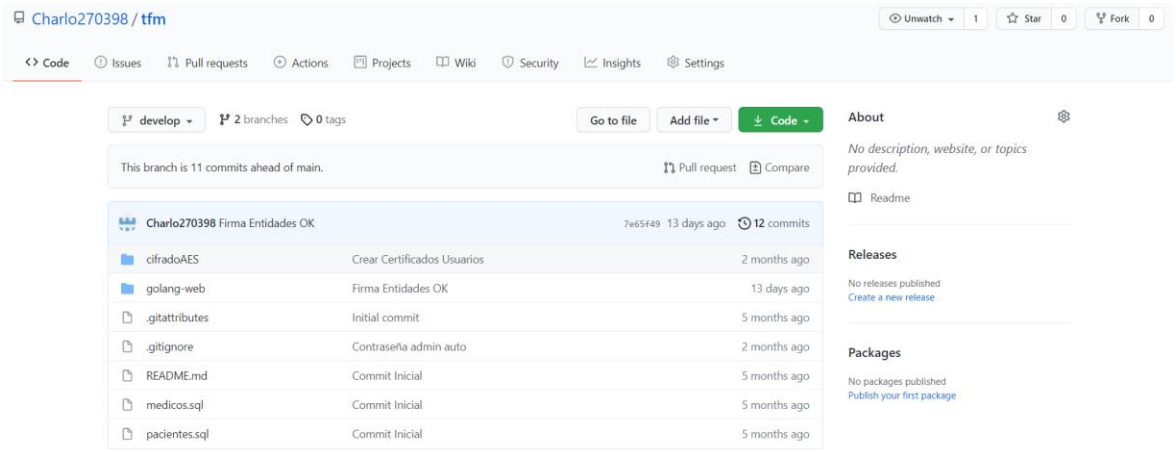

Figura 5.- Repositorio Github empleado en la implementación del proyecto

4.2 Implantación del sistema

Se ha desarrollado un prototipo del sistema que sirve a modo de prueba de concepto para el modelo de gestión segura de datos sensibles propuesto. Entre diversos aspectos, cabe destacar los descritos a continuación.

El sistema implementa la *arquitectura cliente/servidor* con comunicación por TLS (se detalla en la sección 5.1), con la que se encapsula la gestión de los certificados de las entidades en un servidor que actúa como Autoridad Certificadora, la gestión de los datos sensibles de forma segura en el servidor principal de cada entidad médica y la funcionalidad del cliente en cada servidor intermedio, permitiendo que los usuarios finales interactúen con el sistema a través de un navegador web. Esta arquitectura garantiza cierta flexibilidad a la hora de desplegar o escalar el sistema.

El *diseño de datos flexible*¸ que facilita la adaptación del sistema a los posibles casos de uso o flujos de trabajo que puedan aparecer. Como se comenta en la sección 5.2, el sistema hace uso de un sistema de gestión de base de datos relacional, concretamente MySQL, debido a la relativa sencillez de este y a que el sistema es un prototipo. En caso de que se tratara de un sistema más extenso, se podría optar por realizar una migración directa de la base de datos actual a un sistema como Google BigQuery [12] o Amazon Aurora [13], que también hacen uso de un modelo de base de datos relacional, pero con una eficiencia muy superior a la gran mayoría de bases de datos, al ser sistemas pensados para ser utilizados desde un inicio en la nube. Cabe destacar que ceder la gestión de la base de datos a un tercero puede ser algo negativo, ya que la seguridad de todo el sistema pasa a depender de entidades externas. Como

alternativa viable se podría optar por la integración de un *sistema no relacional* escalable para la gestión de datos en los servidores principales de cada entidad médica. Pese a que el diseño de datos está basado en un esquema relacional, existen herramientas que permiten implementar el modelo de datos actual con cambios leves, por ejemplo MongoDB [14].

El *lenguaje de programación* utilizado, Go, que ofrece una amplia variedad de librerías robustas relacionadas con la seguridad y que es un estándar utilizado en el desarrollo de servicios en la nube, siendo compatible de forma nativa con los principales proveedores en este ámbito como Google Cloud, Amazon AES o Microsoft Azure. Además, produce ejecutables estáticos que no requieren de dependencias externas, lo que facilita en gran medida las tareas de instalación y mantenimiento en entornos distribuidos.

A la hora de comercializar el sistema desarrollado, un posible modelo de negocio consiste en la cesión de licencias de uso, que habrían de ser renovadas acorde a los plazos estipulados entre el cliente y el propietario del *software* desarrollado. Cada licencia se vendería en función de una tarifa fija, aunque se ofrecerían servicios extra que incluirían una mayor atención al cliente o el mantenimiento del sistema, garantizando una buena calidad del *software* comprado. El servicio extra debe ser graduable, permitiendo al cliente elegir entre varios grados de mantenimiento, desde un mantenimiento mínimo a un mantenimiento total, variando el precio a pagar en consonancia. Adicionalmente, se pueden ofrecer modificaciones puntuales del sistema para casos concretos, que han de ser presupuestadas e implementadas conforme a los plazos acordados entre cliente y desarrollador.

Se propone Visual Studio Code [15] como entorno de desarrollo para el proyecto, ya que es un *software* sencillo y ampliamente utilizado, además de gratuito. Presenta características muy interesantes, como la integración de extensiones que hacen más fácil la implementación de código, pudiendo añadir extensiones que incorporan autocompletado y comprobación de errores de sintaxis al codificar en Go o JavaScript, entre otros. Además, Visual Studio Code puede sincronizarse con repositorios de control de versiones, agilizando el proceso de integración de código entre desarrolladores en un proyecto.

Durante el desarrollo del proyecto se optaría por utilizar el sistema de control de código fuente GitLab [16] debido a que es gratuito y permite su instalación en la gran mayoría de servidores, posibilitando la ubicación del repositorio en el propio servidor de la empresa desarrolladora, lo que hace el proceso de desarrollo más eficiente y seguro.

En cuanto a los gastos que se pueden producir durante el desarrollo, se han de tener en consideración aspectos como el volumen de la plantilla requerida para llevar a cabo el proyecto, los costes de alquiler y mantenimiento de la oficina, el salario de los trabajadores contratados, la compra de *software* necesario para el proceso de desarrollo, etc. En caso de utilizar un servicio en la nube se debe considerar en el gasto que implicaría su uso, siendo generalmente expresado en dólares por GB por mes; dicho gasto extra debe ser repercutido en la tarifa final aplicada al cliente. Se estima que el proyecto desarrollado debe contar con una inversión inicial cuantiosa, y que será amortizada conforme se vayan vendiendo licencias de uso.

Durante el desarrollo y monitorización del proyecto se seguirá una planificación iterativa e incremental, ligada a los casos de uso del sistema. El plan estará formado por una serie de hitos o entregas, en las que se realizará una pequeña demostración con el cliente para validar

la calidad del *software* desarrollado. En cada iteración se especificará un listado con las tareas que se realizarán durante el *Sprint*, permitiendo que el equipo de desarrollo se autoasigne las tareas a conveniencia, fomentando actitudes como el trabajo en equipo y la autoorganización. Debido a la situación excepcional de pandemia, se debe considerar desde un principio la implantación de políticas que fomenten el teletrabajo, ya que es probable que en cualquier momento se vuelva a producir una merma de la movilidad ciudadana. En adición, existen estudios que avalan la teoría de que los trabajadores son más productivos en entornos favorables al teletrabajo.

5.Desarrollo del proyecto

En este apartado se procede a detallar el diseño del proyecto, enfatizando los aspectos y características relativas a la seguridad de la información.

5.1 Arquitectura del sistema

La arquitectura del sistema consta de cuatro tipos diferentes de nodos: autoridad certificadora, servidor principal, servidor intermedio y cliente JavaScript (navegador web), disponiendo de un único servidor que actúa como Autoridad Certificadora y pudiendo disponer de múltiples instancias de servidores principales, servidores intermedios y clientes JavaScript. Por lo que respecta al esquema de una entidad médica, se considera que está formada por un único servidor principal, uno o varios servidores intermedios y clientes JavaScript.

El esquema básico de una entidad se muestra en la Figura 6, en la que se puede observar la existencia de un nodo central, denominado servidor principal, que ofrece a las entidades médicas un servicio de gestión y almacenaje de datos de forma segura; de los servidores intermedios, que se comunican con el servidor principal a través de una API mediante el uso de un canal seguro; y de los clientes JavaScript, que se comunican con un servidor intermedio específico, también mediante un canal seguro.

Cada servidor intermedio puede ejercer tanto los roles de cliente como de servidor, en función de si resuelve peticiones enviadas por un cliente JavaScript o si realiza peticiones al servidor principal.

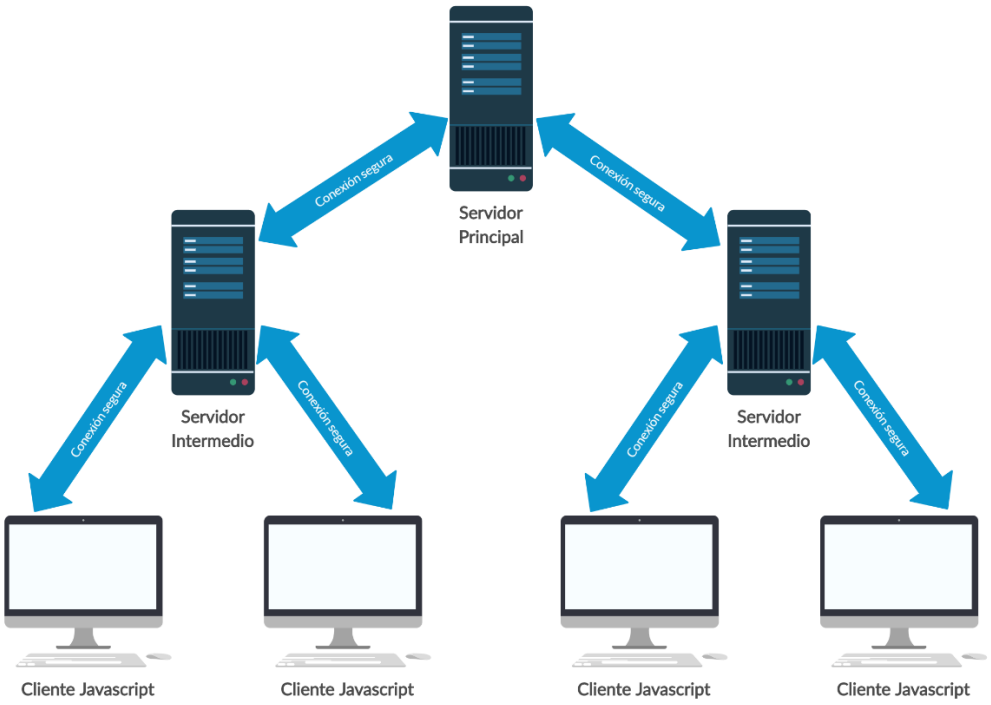

Figura 6.- Esquema básico de la arquitectura de una entidad

El esquema básico de la arquitectura de la autoridad certificadora se muestra en la Figura 6, en la que se puede observar que cada servidor principal de una entidad médica se conecta con la autoridad certificadora, a través de un canal seguro, al igual que todos los servidores principales que forman parte del esquema básico. Las comunicaciones entre entidades se realizan de forma directa, sin necesidad de conectarse con la autoridad certificadora.

Cabe resaltar que esta arquitectura permite añadir nuevas entidades (servidores principales) al esquema de una forma relativamente flexible.

5.2 Diseño de base de datos

Para asegurar la persistencia de datos se ha decidido utilizar una base de datos relacional dado que el proyecto se encuentra en fase de desarrollo y el modelo relacional permite comprender la estructura del esquema de base de datos con relativa facilidad. Además, utilizar un modelo de datos relacional garantiza la integridad referencial, es decir, el borrado de un registro conlleva la eliminación de todos los registros que dependan de él. Esta característica ha sido de gran utilidad en el proyecto, puesto que existe una inmensa cantidad de dependencias dentro del esquema de base de datos, que deben ser controladas. Como se ha descrito en la sección 4.2, se podría sustituir el modelo relacional implementado por uno no relacional en caso de que fuera necesario, ya sea por el aumento del volumen de datos gestionados o la necesidad de escalar el sistema en casos de uso muy concretos.

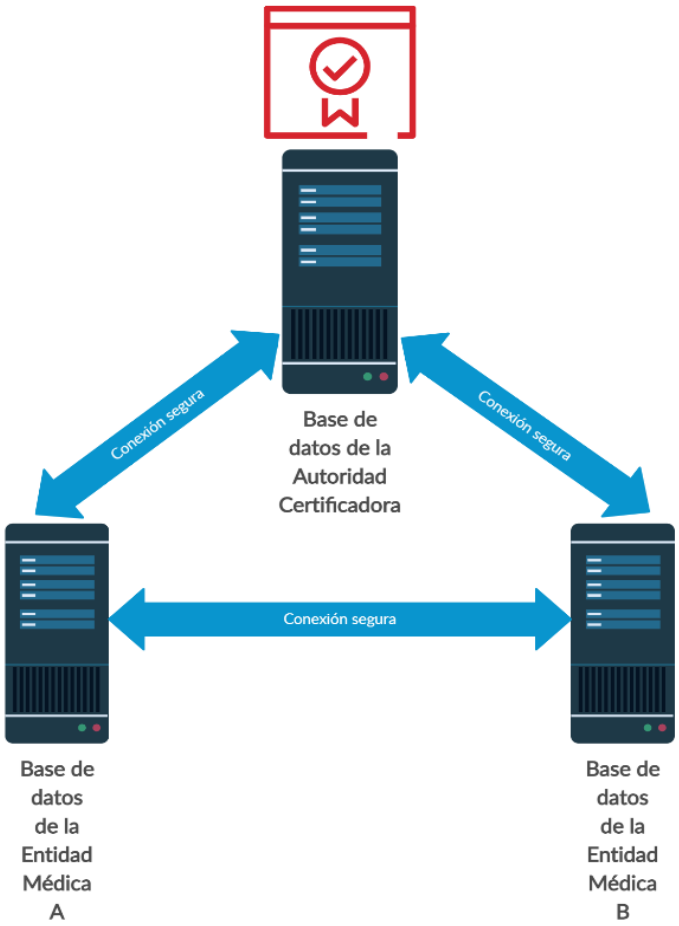

Figura 7.- Comunicación entre los distintos tipos de base de datos en el sistema

Podemos distinguir entre dos tipos de esquema de base de datos en el proyecto, uno para la Autoridad Certificadora que cuenta con un servidor propio y otro para las entidades médicas, que también disponen de un servidor de base de datos.

En todos los servidores del sistema la conexión entre la base de datos y el servidor Go se produce utilizando el *driver* "Go-MySQL-Driver" [17], que proporciona el propio lenguaje Go. Este *driver* permite que Go interactúe con la base de datos a partir de *queries*, de la misma forma que el gestor de base de datos MySQL. El *driver* incluye varias funciones que permiten realizar consultas o ejecuciones a partir de cadenas que representen una sentencia SQL.

En el esquema de base de datos de la Autoridad Certificadora existe una única tabla, denominada "claves_entidades", que contiene la relación entre la dirección IP de una entidad y la clave pública de su certificado. Gracias a esta tabla la Autoridad Certificadora puede determinar si una entidad concreta ya ha sido registrada con anterioridad y compartir la clave pública de su certificado si fuera necesario. En la Figura 12 se muestra el esquema Entidad Relación (ER) de la base de datos de la Autoridad Certificadora.

En cuanto al esquema de base de datos de cada entidad, se ha optado por realizar una división del diagrama ER con el que representar la base de datos debido a que se gestionan una gran cantidad de tablas. Para ello se han agrupado las tablas que tienen un papel importante en las

principales funcionalidades que se han implementado, separando las tablas auxiliares. La partición realizada mantiene como elemento central a la tabla Usuarios, ya que dicha tabla es utilizada en la mayoría de las funciones que han sido implementadas. También se debe diferenciar entre las tablas auxiliares, creadas para la gestión de claves o motivos relacionados con la seguridad; y las tablas principales, que almacenan los datos de las entidades que conforman el sistema. El esquema de base de datos representado como diagrama ER es incluido como apéndice y es descrito brevemente a continuación.

En la Figura 13 se ilustran los atributos que caracterizan a un usuario, que deberá contar al menos con un rol, pudiendo tener varios de forma simultánea. Los usuarios que tengan el rol de paciente disponen de un historial que puede tener entradas y analíticas asignadas, como se detallará más adelante. También constan las citas de un paciente, que deben incluir el identificador del facultativo que atiende dicha consulta, además de su especialidad y clínica asociada. En caso de que un usuario adopte un rol que le haga ejercer como empleado, su nombre completo se insertará en la tabla "empleados_nombres" como texto en claro, ya que estos nombres deben ser accesibles de forma pública.

En la Figura 14 se exponen las tablas utilizadas durante el proceso de solicitud de permisos entre los empleados y pacientes. Un empleado autorizado puede inscribir solicitudes de acceso con varios niveles de granularidad: acceso a los datos básicos del historial, a una entrada concreta, a una analítica concreta o al historial completo. Si se solicita el acceso a los datos básicos o al historial no es necesario almacenar un identificador del historial para completar el proceso de solicitud, ya que cada paciente tiene asociada una única historia clínica. En cambio, si se solicita el acceso a una entrada o analítica concretas se requiere el identificador del historial al que están asociadas.

En la Figura 15 se muestran las tablas involucradas en la compartición de claves de los historiales almacenados en el sistema. A grandes rasgos, existen tres tipos de claves a compartir: la clave que cifra los datos básicos del historial, la clave que cifra una analítica y la clave que cifra una entrada. Para cada tipo de clave existe una tabla propia en la que se almacenan las claves compartidas.

En la Figura 16 se ilustran las tablas empleadas en el proceso de autenticación del usuario, la gestión de las claves RSA y la gestión de los certificados. Existe una tabla en la que se almacenan las parejas de claves RSA de los usuarios, otra en la que se almacenan las parejas de claves RSA de los certificados de los usuarios y otra en la que se almacena la pareja de claves RSA del sistema. Cada usuario posee un *token* con una fecha de expiración asignada que le permite ejecutar acciones en el servidor principal de la entidad. También se almacena el *hash* SHA-512 que resume el DNI de cada usuario, necesario para realizar búsquedas de usuarios por DNI.

En la Figura 17 se exponen las tablas en las que se guardan los datos de las analíticas de forma anonimizada, siendo almacenados como texto en claro. Cada analítica es identificada por un valor aleatorio que la diferencia del resto y puede ser relacionada con uno o más *tags*, que son definidos en otra tabla.

5.3 Diseño de la capa de seguridad

En los siguientes subapartados se detallan los distintos componentes que conforman la capa de seguridad del sistema.

5.3.1 Herramientas criptográficas

En este apartado se describen resumidamente las diversas herramientas criptográficas utilizadas para gestionar la seguridad del sistema desarrollado.

Se ha empleado el protocolo de transporte seguro *Transport Layer Security*, TLS [18], para proporcionar seguridad a las comunicaciones entre Autoridad Certificadora, servidor principal, servidor intermedio y usuarios. TLS es un protocolo ampliamente utilizado en correo electrónico, navegación web, mensajería instantánea o voz sobre IP.

Se ha utilizado criptografía simétrica a la hora de cifrar los datos, eligiendo un algoritmo de cifrado en bloque ampliamente reconocido y que cuenta con excelentes propiedades de eficiencia y seguridad, *Advanced Encryption Standard* o AES [19]. Dicho algoritmo cuenta con tres grados de seguridad en función del tamaño de clave seleccionado: 128, 192 y 256 bits, conociéndose como AES128, AES192 y AES256 respectivamente. Se ha optado por utilizar únicamente AES256 en el proyecto desarrollado.

La criptografía asimétrica ha sido utilizada para cifrar y compartir claves, habiendo seleccionado uno de los algoritmos más extensamente utilizados: RSA [20]. RSA requiere de una pareja de claves, a diferencia de AES; una de ellas es pública y es empleada en el proceso de cifrado mientras que la otra es privada y es empleada en el proceso de descifrado. La clave privada se puede obtener a partir de la pública conociendo unos determinados valores, siendo un cálculo muy complejo computacionalmente al que se enfrenta un atacante que quiera sustraer esos valores y, a partir de estos, obtener la clave privada con la que es posible descifrar.

En el contexto del sistema desarrollado, se ha optado por elegir un tamaño de clave para RSA de 2048 bits, considerado como seguro para la capacidad de cómputo actual. La criptografía asimétrica también ha sido utilizada para generar certificados en el sistema, empleando el formato estándar X.509 [21], que permite a los titulares firmar documentos o autenticarse, entre otras aplicaciones. Dichos certificados asocian identidades, como usuarios o sitios web, a un par de claves RSA de forma segura. En el sistema desarrollado, los certificados son utilizados tanto para realizar conexiones a través de un canal seguro como para identificar a los usuarios, entidades médicas y a la propia Autoridad Certificadora.

Las funciones *hash* asignan a cualquier registro digital una secuencia binara con longitud fija mediante una correspondencia unívoca [22]. En el ámbito de la seguridad de la información cumple un papel importante en la verificación de la integridad de la misma permitiendo su uso el no almacenamiento en claro de las contraseñas de los usuarios y evitando, de este modo, su sustracción o la posible accesibilidad en el sistema. Para nuestro sistema se ha optado por la utilización de la función *hash* estándar del NIST, SHA3, con un resumen de 512 bits, representada en esta memoria y en la literatura como $\mathrm{SHA3}_{512}$.

Se puede atacar la tabla que almacena los *hashes* de las contraseñas de los usuarios para intentar obtener algunas contraseñas (o el total) mediante lo que se conoce como ataque de

tablas precalculadas o arcoíris (*rainbow tables*) que consiste en precalcular el resumen de una serie, más o menos, exhaustiva de posibles contraseñas y comparar con los valores almacenados. Una posible solución a este ataque consiste en forzar a que las tablas precalculadas sean de tamaños muy grandes (prácticamente intratables) haciendo muy costosa la comparación. Añadiendo aleatoriedad a las contraseñas se consigue este efecto y en la práctica se utiliza lo que se conoce como sal [23], consistente en una cadena de bits generada aleatoriamente que es concatenada con la contraseña elegida por el usuario para, posteriormente, obtener el resumen de toda la cadena.

La solución mencionada anteriormente no es suficiente dado el incremento de la capacidad computacional de las GPU (*Graphics Processing Unit*) que permite la comparación, mediante lo que se denomina ataque exhaustivo o por fuerza bruta, de todas las posibles contraseñas de un determinado tamaño en un tiempo prudencial. Una forma de evitar el ataque descrito consiste en la utilización de las denominadas funciones de derivación de clave basada en contraseña o PBKDF (*Password Based Key Derivation Function*) [15], que son funciones *hash* más costosas computacionalmente y configurables en base a parámetros como el tiempo de computación, la memoria utilizada y el número de hilos. En implementación del sistema se ha optado por Argon2id [24], que es una función de derivación de clave elegida como ganadora del concurso *Password Hashing* [25] de 2015. Dicha función incluye la generación de sal para evitar ataques mediante tablas arcoíris, como se ha descrito anteriormente.

5.3.2 Canal Seguro

Es muy importante garantizar que la conexión entre cada uno de los nodos que existen en el sistema desarrollado se produzca a través de un canal seguro, protegiéndolo de posibles ataques como *sniffing*, *man in the middle*, etc. Cada vez es más importante para las empresas y entidades garantizar la seguridad en las conexiones, pudiendo ser un aspecto crítico en algunos modelos de negocio, principalmente los que impliquen la gestión de datos sensibles.

Para establecer conexiones seguras entre los nodos del sistema implementado se ha decidido utilizar la librería crypto/tls de Go, que implementa las versiones 1.2 y 1.3 del protocolo de comunicación segura TLS (*Transport Layer Security*). Esta librería permite establecer un puerto de escucha que utiliza dicho protocolo a partir de una clave y un certificado.

Se ha optado por utilizar certificados auto firmados, ya que el proyecto se encuentra en una fase de desarrollo. La herramienta utilizada para generar dichos certificados ha sido *openssl*, que permite generar certificados auto firmados de forma relativamente sencilla y exportarlos en varios formatos. Cada nodo del sistema cuenta con un certificado distinto, generando un certificado/clave para el servidor de Autoridad Certificadora, otro por cada servidor principal de cada entidad médica y otro por cada servidor intermedio. Los pares certificado/clave se ubican en la carpeta raíz de los servidores de cada nodo del sistema y se corresponden con los archivos "cert.pem" y "key.pem" respectivamente.

En resumen, TLS nos aporta una herramienta útil para garantizar comunicaciones seguras entre los nodos del sistema, permitiendo el intercambio de datos de una forma segura y privada entre cada uno de ellos. En definitiva, el uso de esta herramienta evita posibles ataques como, por ejemplo, la sustracción de datos cuando se realiza una petición REST de tipo POST.

5.3.3 Notación del cifrado

Con el fin de facilitar la comprensión y simplificar la descripción, se plantea la siguiente notación:

$F_s(x)$ = primeros s bits de x

$L_s(x)$ = últimos s bits de x

pwd_e = contraseña del elemento e (usuario, entidad médica, autoridad certificadora, ...) del sistema

pK_{u_i} = clave pública RSA del usuario u_i

pk_{u_i} = clave privada RSA del usuario u_i

Cpk_{u_i} = clave privada RSA del usuario u_i cifrada con AES

pK_s = clave pública RSA del sistema

pk_s = clave privada RSA del sistema

$D_k^a(c)$ = descifrado mediante el algoritmo a con la clave k de la cadena c

$E_k^a(m)$ = cifrado mediante el algoritmo a con la clave k de la cadena m

R = registro descifrado

k_R = clave AES del registro R, por lo general, aleatoria

CR = registro cifrado con AES

h(m) = $SHA3_{256}(m)$ = resultado de aplicar $SHA3_{256}$ a la cadena m

A(m)=Argon2id(m) = resultado de aplicar Argon2id a la cadena m

crt_{AC} = certificado raiz de la Autoridad Certificadora

pKc_{AC} = clave pública RSA vinculada al certificado raiz de la Autoridad Certificadora

pkc_{AC} = clave privada RSA vinculada al certificado raiz de la Autoridad Certificadora

crt_{EM_i} = certificado de la Entidad Médica i

pKc_{EM_i} = clave pública RSA vinculada al certificado de la Entidad Médica i

pkc_{EM_i} = clave privada RSA vinculada al certificado de la Entidad Médica i

crt_{u_i} = certificado del usuario i

pKc_{u_i} = clave pública RSA vinculada al certificado del usuario i

pkc_{u_i} = clave privada RSA vinculada al certificado del usuario i

5.3.4 Puesta en marcha de la autoridad certificadora

A la hora de iniciar el sistema desde cero, en primer lugar se debe poner en marcha la autoridad certificadora ejecutando su servidor en el que se comprueba si ya existe un

certificado raíz, crt_{AC}, y su clave privada asociada cifrada con AES, $E^{AES256}_{h(pwd_{AC})}(pkc_{AC})$. Si tras la comprobación se detecta que el servidor no tiene almacenado un certificado raíz, junto con su clave privada RSA asociada cifrada con AES, se procede a su generación.

Para generar el certificado y la clave privada asociada cifrada con AES se solicita al administrador, inicialmente, una contraseña que cumpla unas restricciones de complejidad básicas, pwd_{AC}, y se calcula su resumen SHA256, $h(pwd_{AC})$; a continuación, se genera un par de claves RSA, pKc_{AC}, pkc_{AC} y, a partir de estas, se crea un certificado estándar X.509 cuya estructura requiere de los datos básicos proporcionados en el fichero de configuración (organización, país, provincia, localidad, código postal, etc.). Una vez generado se cifra la clave privada RSA con AES y clave $h(pwd_{AC})$,

$$E^{AES256}_{h(pwd_{AC})}(pkc_{AC}),$$

almacenándose el resultado en la base de datos junto con el certificado raíz, crt_{AC}.

Una vez comprobado que estos dos elementos, el certificado crt_{AC} y la clave privada RSA cifrada con AES, se encuentran almacenados en el servidor se solicita al administrador la contraseña para descifrar la clave privada RSA con AES y clave $h(pwd_{AC})$

$$D^{AES256}_{h(pwd_{AC})}\left(E^{AES256}_{h(pwd_{AC})}(pkc_{AC})\right) = pkc_{AC}.$$

Cuando se obtiene pkc_{AC} se procede a comprobar su autenticidad mediante la firma de un mensaje aleatorio, *nonce*,

$$D^{RSA}_{pkc_{AC}}(nonce)$$

y la posterior verificación de la firma con la clave pública del certificado raíz almacenado en la base de datos de la autoridad certificadora

$$E^{RSA}_{pKc_{AC}}\left(D^{RSA}_{pkc_{AC}}(nonce)\right) = nonce$$

Si la verificación se lleva a cabo satisfactoriamente, la clave privada de la autoridad certificadora, pkc_{AC}, es almacenada en memoria y será utilizada en la firma de los nuevos certificados generados a las entidades médicas que así lo soliciten a través de la API.

Una vez realizado el proceso anterior (certificado raíz, contraseña, clave privada, etc.) se inicia el servidor HTTP de la autoridad certificadora en el que se ha habilitado comunicación segura mediante TLS con las entidades médicas que se conecten.

En la Figura 8 se muestra un pseudocódigo del proceso descrito con anterioridad.

```
//El proceso es realizado en la autoridad certificadora

1. Se comprueba si la autoridad certificadora ya está
   utilizando un certificado raíz. En caso afirmativo se
   salta al paso 8.
2. Se solicita la contraseña y se calcula su hash SHA256.
3. Se genera un par de claves RSA, utilizadas en el
   certificado.
4. Se genera un certificado con estándar X.509 a partir
   del par de claves RSA y los datos proporcionados en la
   configuración.
5. Se cifra la clave privada RSA del certificado con AES,
   utilizando como clave la cadena obtenida en el paso 2.
6. Se almacena el certificado y la clave privada RSA
   cifrada.

ERROR DURANTE EL PROCESO DE GENERACIÓN DEL CERTIFICADO
7. Se elimina el certificado y la clave privada RSA
   almacenada en el paso 6 y regreso al paso 2.

GENERACIÓN DEL CERTIFICADO REALIZADA CORRECTAMENTE
8. Se solicita la contraseña y se calcula su hash SHA256.
9. Se descifra la clave privada RSA con AES utilizando
   como clave la cadena obtenida en el paso 8.

10. Se verifica la autenticidad de la contraseña mediante
    el proceso de firma digital con la clave privada
    obtenida y la pública del certificado.

  VERIFICACIÓN DE CONTRASEÑA INCORRECTA
  11. Se termina la ejecución.

  VERIFICACIÓN DE CONTRASEÑA CORRECTA
  12. Se almacena la clave privada RSA en memoria.
```

Figura 8.- Algoritmo de puesta en marcha de la autoridad certificadora

5.3.5 Registro de una entidad en la autoridad certificadora

Cuando una entidad médica inicia su servidor principal se comprueba si ya dispone de un certificado que represente a la entidad almacenado en su base de datos, además de la clave privada perteneciente a ese certificado, cifrada. En caso negativo se solicita una contraseña, se calcula su hash SHA3-256, se genera un par de claves RSA y, utilizando la clave privada, se genera un certificado X509 que requiere de datos que lo identifiquen (organización, país, provincia, localidad, código postal, etc.).

Cada entidad médica, EM_i, dispone de un fichero de configuración en el que se detallan los datos indicados anteriormente de manera que su certificado sea generado en base a ellos. Tras generar el certificado se procede a cifrar la clave privada con AES256 y clave el hash obtenido a partir de la contraseña,

$$E^{AES256}_{h\left(pwd_{EM_i}\right)}\left(pkc_{EM_i}\right),$$

almacenando el resultado en la base de datos de la entidad médica, EM_i, junto con el certificado.

Cuando se comprueba que ya existen tanto el certificado como la clave privada asociada al mismo, se solicita una contraseña, se calcula su hash SHA3-256 y se descifra la clave privada con AES256 y clave el hash,

$$D^{AES256}_{h\left(pwd_{EM_i}\right)}\left(E^{AES256}_{h\left(pwd_{EM_i}\right)}\left(pkc_{EM_i}\right)\right)=pkc_{EM_i}.$$

Una vez descifrada la clave privada, EM_i, firma un mensaje aleatorio, *nonce*, con la misma

$$D^{RSA}_{pkc_{EM_i}}\left(nonce\right)$$

y verifica la firma con la clave pública del certificado almacenado en la base de datos de la entidad médica

$$E^{RSA}_{pKc_{EM_i}}\left(D^{RSA}_{pkc_{EM_i}}\left(nonce\right)\right)=nonce.$$

Si la verificación no es correcta se solicita de nuevo la contraseña y si lo es se almacena la clave privada en memoria y se comprueba que tanto el certificado raíz de la autoridad certificadora como el certificado de la entidad médica ya firmado por la autoridad certificadora estén almacenados en el servidor de EM_i.

En el caso de que ambos certificados no se encuentren almacenados, se envía una petición a la autoridad certificadora, en la que se adjunta una copia de la parte pública del certificado de la entidad médica. La autoridad certificadora recibe la solicitud y comprueba si en la tabla que correlaciona direcciones IP y certificados ya existe una entrada en la base de datos con la IP origen de la solicitud, o bien si existe un certificado idéntico al enviado. Si no hay un registro previo, se almacena el certificado de la entidad junto con su dirección IP y, posteriormente, el certificado es firmado y se envía como respuesta junto con la parte pública del certificado raíz de la autoridad certificadora. En la Figura 9**¡Error! No se encuentra el origen de la referencia.** se muestra de forma resumida el proceso descrito anteriormente.

Una vez la entidad médica recibe respuesta a su solicitud, almacena su certificado firmado y el certificado raíz de la autoridad certificadora en base de datos, ya que serán necesarios

en el proceso de compartición de información entre entidades médicas que se detalla en el apartado 5.3.7.

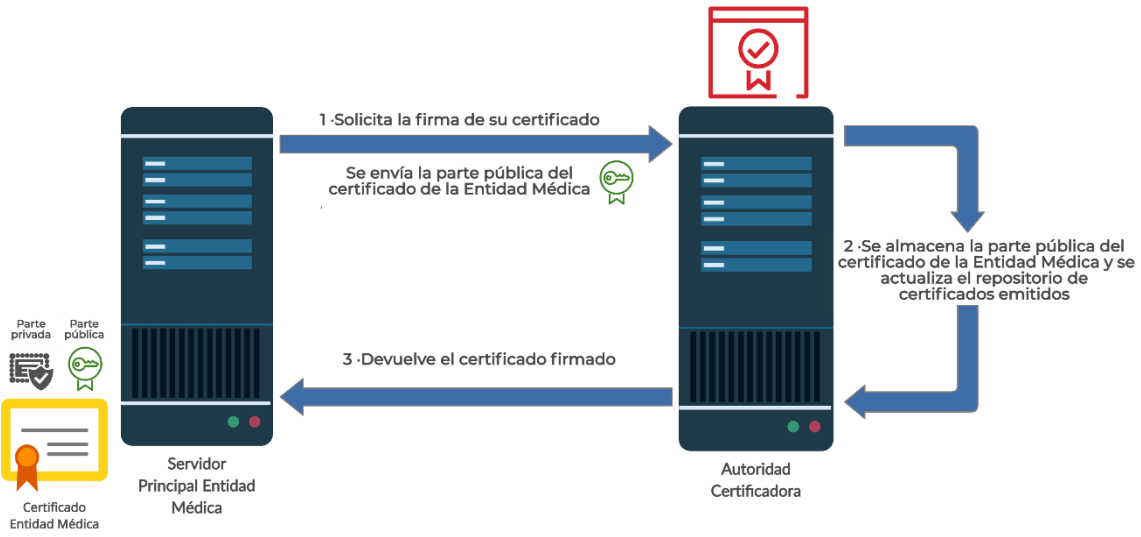

Figura 9.- Esquema de la comunicación para el registro de una entidad médica en el sistema

5.3.6 Creación de usuario en una entidad

El sistema permite crear usuarios mediante un formulario de registro abierto en la página inicial, en el que un usuario se puede dar de alta como paciente. En este formulario se debe especificar, obligatoriamente, algunos datos básicos como nombre, apellidos, DNI, correo electrónico, etc. además de otros datos que son opcionales. También existe la posibilidad de que un usuario sea creado por un Administrador del sistema, que debe rellenar un formulario de registro con los mismos datos básicos enumerados anteriormente e indicar los roles que podrá adoptar el nuevo usuario: paciente, enfermería, facultativo, administrador, administrador de sistema, etc. Pese a que existen dos formas para dar de alta a un usuario, la lógica que permite que esto ocurra es común en ambos procesos.

Cuando se completa el formulario del usuario u_i, este es enviado desde el navegador al servidor intermedio, en el que se calcula el *hash* SHA3 de 512 bits de la contraseña proporcionada, $\mathrm{SHA3}_{512}\left(\mathrm{pwd}_{u_i}\right)$, y se genera un par de claves RSA para u_i, pK_{u_i} y $\mathbf{pk}_{u_i}$. Tras generar la pareja de claves RSA la clave pública se mantiene sin cifrar, mientras que la privada es cifrada en el servidor intermedio utilizando AES256, siendo la clave de cifrado los 256 bits menos significativos obtenidos mediante el *hash* calculado sobre la contraseña proporcionada, $\mathrm{L}_{256}\left[\mathrm{h}\left(\mathrm{pwd}_{u_i}\right)\right]$, esto es

$$\mathrm{E}^{\mathrm{AES256}}_{\mathrm{L}_{256}\left[\mathrm{h}\left(\mathrm{pwd}_{u_i}\right)\right]}\left(\mathrm{k}_{u_i}\right);$$

enviando el resultado al servidor principal de la entidad, donde es almacenado en la base de datos.

El resto de los datos aportados en el formulario son cifrados en el servidor intermedio utilizando AES256 con una clave aleatoria, $k_{R_{u_i}}$, que, por seguridad, es cifrada con la clave pública del usuario u_i y la clave pública del sistema,

$$E^{RSA}_{K_{u_i}}\left(k_{R_{u_i}}\right) \text{ y } E^{RSA}_{K_s}\left(k_{R_{u_i}}\right).$$

Adicionalmente, se calcula el *hash* SHA3 de 512 bits del DNI del usuario u_i, $SHA3_{512}\left(DNI_{u_i}\right)$, ya que este valor será necesario a la hora de realizar búsquedas de usuarios.

Una vez realizado todo el proceso descrito anteriormente se envían los registros y claves cifradas al servidor principal, a través de un canal seguro, adjuntando los primeros 256 bits del *hash* de la contraseña, $F_{256}\left[h\left(pwd_{u_i}\right)\right]$.

Una vez que el servidor principal ha recibido la petición de registro se define la clave de inicio de sesión del usuario, aplicando la función de derivación de clave Argon2Id sobre los 256 primeros bits del *hash* de la contraseña,

$$Argon2ID\left(F_{256}\left[h\left(pwd_{u_i}\right)\right]\right),$$

y se genera un par de claves RSA, pKc_{u_i}, pkc_{u_i}, a partir de las cuales se crea un certificado estándar X.509 cuya estructura requiere de los datos básicos proporcionados en el fichero de configuración del servidor principal (organización, país, provincia, localidad, código postal, etc.) y su identificador es correlacionado con el DNI del usuario en la base de datos. El certificado es firmado con la clave privada del certificado del servidor principal de la entidad,

$$D^{RSA}_{pkc_{EM_i}}\left(crt_{u_i}\right).$$

Tras crear el certificado del usuario u_i, se procede a cifrar la clave privada de su certificado con AES256 y una clave aleatoria $k_{pkc_{u_i}}$,

$$E^{AES256}_{k_{pkc_{u_i}}}\left(pkc_{u_i}\right),$$

y, posteriormente, a cifrar la clave aleatoria $k_{pkc_{u_i}}$ con la clave pública del usuario y la clave pública del sistema,

$$E^{RSA}_{K_{u_i}}\left(k_{pkc_{u_i}}\right) \text{ y } E^{RSA}_{K_s}\left(k_{pkc_{u_i}}\right).$$

Finalmente, se añade a la base de datos la información básica relativa al usuario ya cifrada, así como el *hash* de su clave de inicio de sesión, la clave pública de su certificado, la clave privada de su certificado cifrada, las claves aleatorias cifradas con RSA y el par de claves RSA del usuario mencionadas con anterioridad.

Una vez que el usuario es añadido satisfactoriamente a la base de datos, se genera un *token* aleatorio, que consiste en una cadena de 128 bits, al que se asigna un periodo de expiración de 30 minutos. Cada vez que el usuario pretenda realizar una petición al servidor principal

deberá aportar el *token*, que será validado y renovado siempre y cuando la validación sea satisfactoria.

Si el proceso de inserción en la base de datos del usuario finaliza sin ningún incidente el *token y* su periodo de expiración son almacenados en la base de datos, y se devuelve una respuesta al servidor intermedio adjuntando el identificador del usuario y el *token*. Cuando el servidor intermedio recibe la respuesta se crea un objeto de sesión que contiene el identificador y *token* devueltos.

En la Figura 10 se muestra un pseudocódigo del proceso anterior.

5.3.7 Solicitud de información a una entidad externa

El sistema permite que un facultativo, u_f, de una entidad médica EM_i pueda solicitar o añadir información de cualquier paciente, u_p, registrado en cualquiera de las entidades adheridas al sistema. Para ello el facultativo accede a un panel en el que, en primer lugar, debe seleccionar la entidad médica EM_j en la que se ubica el paciente que busca y, en segundo lugar, debe indicar el DNI del paciente en cuestión.

El servidor intermedio recibe la petición en la que se adjuntan estos dos parámetros y realiza una petición al servidor principal de la entidad EM_i para recuperar la clave privada del certificado del facultativo y su clave pública asociada, descifrando la clave privada con AES256 y clave los 256 bits menos significativos del *hash* de la contraseña del facultativo almacenados en el objeto de sesión

$$D^{AES256}_{L_{256}\left[h\left(pwd_{u_f}\right)\right]}\left(E^{AES256}_{L_{256}\left[h\left(pwd_{u_f}\right)\right]}\left(pkc_{u_f}\right)\right) = pkc_{u_f}.$$

Una vez descifrada la clave privada RSA, se procede a firmar con la misma el resumen obtenido tras aplicar el *hash* SHA3-512 sobre el objeto que contiene el par entidad médica/paciente de la solicitud (EM_j, u_p). A continuación, se envía una petición al servidor principal de EM_i conteniendo el par entidad médica/paciente, el *hash* firmado y la clave pública del certificado del facultativo. El servidor principal verifica que el certificado del facultativo ha sido emitido por EM_i y, en caso afirmativo, se reenvía la petición al servidor principal de EM_j adjuntando el certificado de EM_i firmado por la Autoridad Certificadora, y el certificado de u_f firmado por EM_i.

```
//El proceso es realizado en el servidor intermedio

1·Sobre el DNI se calcula el hash SHA3-512.
2·Sobre la contraseña se calcula el hash SHA3-512.
3·El par de claves RSA del usuario es generado.
4·Las claves privadas son cifradas con AES utilizando
  los 256 bits menos significativos del hash del paso 1.
5·Se genera una clave aleatoria de 256 bits.
6·Los datos personales son cifrados con AES utilizando
  la clave del paso 5.
7·La clave del paso 5 es cifrada con la clave pública
  del usuario, obtenida en el paso 3.
8·La clave del paso 5 es cifrada con la clave pública
  del sistema.
9·Se envía una petición al servidor principal
  adjuntando los pares de claves RSA, los datos
  cifrados, el hash del paso 1 y los 256 bits más
  significativos del hash del paso 2, esperando
  respuesta.

//El proceso es realizado en el servidor principal
10·El par de claves RSA del certificado del usuario es
   generado.
11·Se genera el certificado del usuario a partir del
   par de claves del paso 10.
12·Se genera una clave aleatoria de 256 bits.
11·La clave privada del paso 10 es cifrada con AES y la
   clave del paso 12.
12·La clave del paso 12 es cifrada con la clave pública
   del usuario.
13·La clave del paso 12 es cifrada con la clave pública
   del sistema.
14·Se crea un token de sesión.
15·Se insertan todos los datos cifrado, claves y token
   junto con su tiempo de expiración.

//El proceso es realizado en el servidor intermedio
RESPUESTA INVÁLIDA
   16·Se muestra un mensaje de error.
RESPUESTA VÁLIDA
   16·Se crea un objeto de sesión almacenando en su
      interior el token e identificador del usuario,
      recibidos en el paso 9 y la segunda mitad del
      hash del paso 2.
```

Figura 10.- Algoritmo del proceso de registro de un usuario en el sistema

Cuando EM_j recibe la solicitud, en primer lugar, verifica que el certificado de EM_i ha sido firmado por la Autoridad Certificadora. Para ello se cifra el certificado de EM_i, firmado por la Autoridad Certificadora, con RSA y la clave pública del certificado raíz de la Autoridad Certificadora,

$$E^{RSA}_{pKc_{AC}}\left(D^{RSA}_{pkc_{AC}}\left(pKc_{EM_i}\right)\right) = pKc_{EM_i}\,.$$

Si la verificación es incorrecta se envía una petición a la Autoridad Certificadora en la que se adjuntan datos que describen la incidencia y que serán almacenados en el servidor de la Autoridad Certificadora a modo de *logs*.

En caso de que la verificación se lleve a cabo correctamente, se procede a confirmar que el certificado del facultativo u_f ha sido firmado por la entidad EM_i. Para ello se cifra el certificado de u_f, firmado por la entidad EM_i, con RSA y la clave pública del certificado de la entidad EM_i,

$$E^{RSA}_{pKc_{EM_i}}\left(D^{RSA}_{pkc_{EM_i}}\left(pKc_{u_f}\right)\right) = pKc_{u_f}\,.$$

De nuevo, si la verificación es incorrecta se envía una petición a la Autoridad Certificadora en la que se adjuntan datos que describen la incidencia, siendo almacenados como *logs* y si es correcta se devuelve la información solicitada, ya descifrada. En la Figura 11 se resume el proceso de comunicación realizado entre todos los nodos involucrados.

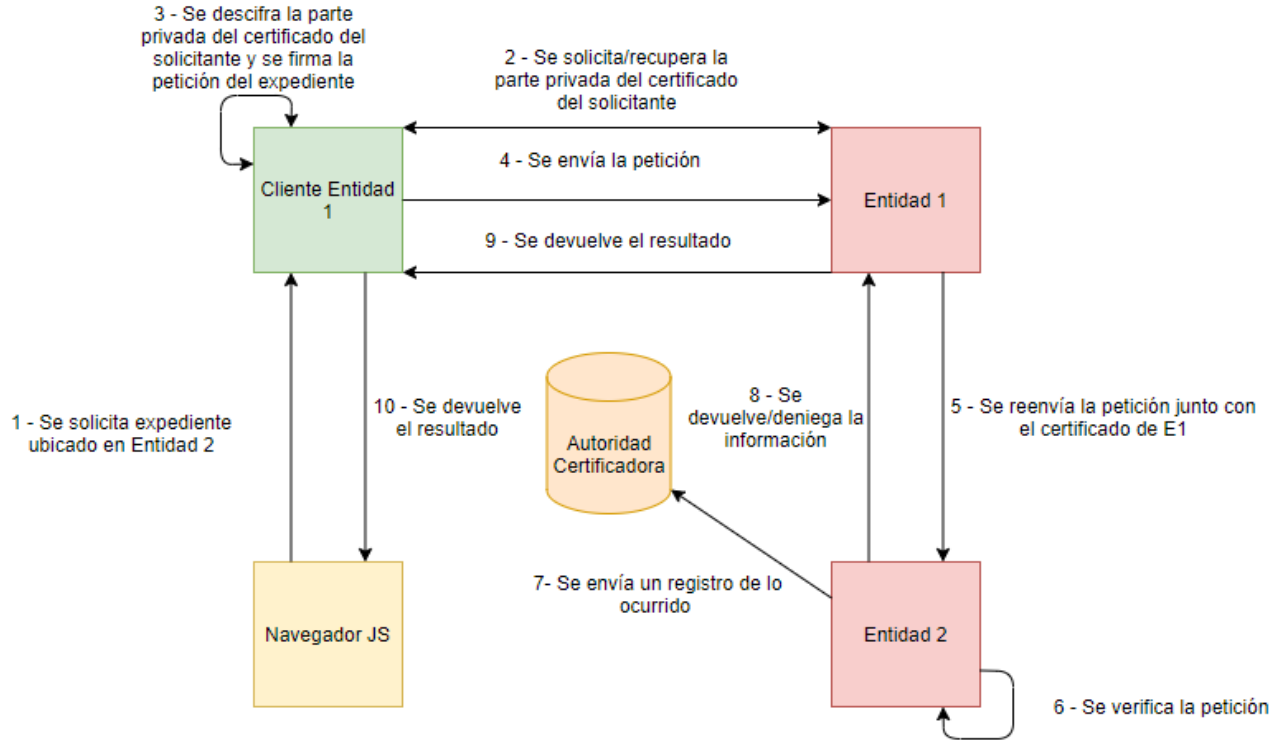

Figura 11.- Esquema de comunicación de la solicitud de información a una entidad externa

5.3.8 Librerías utilizadas

Este subapartado contiene una breve descripción de las librerías usadas durante la implementación del proyecto, indicando las modalidades de cifrado utilizadas y otras técnicas empleadas.

Una gran ventaja del lenguaje utilizado en la implementación de la aplicación, Go, es la gran variedad de librerías relacionadas con la criptografía y las funciones que estas incluyen. Gracias a estas librerías se han podido implementar la mayoría de las técnicas de ciberseguridad necesarias para el funcionamiento del sistema desarrollado en el proyecto, destacando entre otras:

- **crypto/x509** [26]: Esta librería permite analizar claves y certificados en el formato estándar X.509. Entre las funciones proporcionadas por la librería se encuentran CreateCertificate, usada a la hora de generar certificados, y MarshalPKCS1PrivateKey, que permite convertir una clave privada RSA a formato PKCS1.
- **crypto/rsa** [27]: La librería implementa el cifrado RSA, utilizado de forma adaptada en el proyecto. Las funciones que han sido adaptadas permiten generar parejas de claves RSA, cifrar cadenas de texto con clave pública y descifrarlas con la clave privada. Adicionalmente se han implementado funciones que permiten transformar

las claves RSA a *arrays* de bytes codificados en base 64 [28] y viceversa, ya que es más que recomendable almacenar las claves RSA en base 64, evitando posibles problemas de compatibilidad entre codificaciones.

- **crypto/argon2** [29]: Implementa la función IDKey, que permite implementar la función de derivación de clave Argon2id. La función requiere de varios parámetros: una cadena que contenga la contraseña a almacenar, un entero que especifique el tamaño de la sal, un *array* de bytes que defina la sal y varios parámetros que configuren el coste computacional que requiere la operación (como el número de hilos, tiempo, o uso de memoria en kibibytes). La función devuelve un *array* de bytes que representa la clave. El manual de uso de Argon2 especifica como recomendable la utilización de 64MB de memoria y que el número de pasadas sobre memoria o tiempo sea igual a 1, siendo también recomendable obtener una sal generada de forma aleatoria de un tamaño de 16 bytes.
- **crypto/aes** [30]**:** La librería implementa el algoritmo de cifrado AES (versión reducida de Rijndael) conforme a las normas federales de procesamiento de información de los Estados Unidos (FIPS). Entre las funciones proporcionadas podemos destacar NewCipher, que permite crear un nuevo cifrador a partir de la clave que se introduzca como parámetro. La clave debe tener un tamaño de 16, 24 o 32 bytes, siendo AES128, AES192 o AES256 el cifrado utilizado, en concordancia con el tamaño elegido. Se puede indicar el modo de cifrado de bloque con la librería "crypto/cipher", siendo utilizado el modo CFB para el proyecto.
- **crypto/cipher** [31]**:** Provee los modos de cifrado de bloque estándar a utilizar en implementaciones de cifrado de bloque a bajo nivel.
- **crypto/sha256** [32]**:** Esta librería implementa varios algoritmos *hash*, entre ellos SHA224 y SHA256, todos ellos acordes a la definición establecida en FIPS 180-4.
- **crypto/sha512** [33]**:** Esta librería implementa varios algoritmos *hash*, entre ellos SHA-384, SHA-512, SHA-512/224 y SHA-512/256, todos ellos acordes a la definición establecida en FIPS 180-4. En el proyecto se utiliza la función SHA-512 para calcular el resumen de la información que va a ser firmada y para generar el resumen de la contraseña que especifican los usuarios, utilizado en varios procesos del sistema.
- **encoding/base64** [34]**:** El paquete implementa la codificación en base 64 especificada en la RFC 4648. Incluye funciones que permiten codificar y

descodificar, utilizadas antes y después de cifrar las claves que son gestionadas en el proyecto.

5.4 Problemas abiertos

En este subapartado se plantean algunas cuestiones, surgidas durante el desarrollo del proyecto, que cabría resolver en caso de implantar el sistema en un entorno real.

Una de las problemáticas del sistema actual, y de cualquier sistema que haga uso de una Autoridad Certificadora, es la instalación del certificado raíz en cada una de las entidades médicas que confíen en la Autoridad Certificadora. Las entidades médicas deben obtener el certificado raíz a través de una fuente fiable y verificar que, efectivamente, el certificado es real y se corresponde con el certificado raíz de la Autoridad Certificadora en cuestión, ya que sería posible realizar un ataque *Man in the middle* o ataque de intermediario mediante una suplantación de este. Se proponen dos soluciones: abrir la posibilidad de instalar el certificado raíz en cada entidad de forma física, por ejemplo, un técnico responsable debería transferir el certificado al servidor principal de cada entidad de forma presencial; o también se puede optar por establecer como certificado raíz a un certificado firmado por una entidad superior, que ha de ser firmado por un proveedor de confianza. Existe una amplia lista de proveedores de certificados de confianza, como IdenTrust, DigiCert, GoDaddy o GlobalSign.

Otra problemática a resolver es la caducidad o revocación de las claves y certificados gestionados por el sistema, ya que la implementación no contempla que un certificado pueda ser revocado puntualmente debido a eventos como la baja de un usuario, el robo de credenciales, órdenes judiciales, etc. La solución propuesta consiste en implementar una lista de revocación de certificados o CLR, ubicada en el servidor de la Autoridad Certificadora, que ha de ser pública para que cualquier usuario pueda consultar dicha CLR y comprobar la validez de su certificado.

También se ha de tener en cuenta la implementación de sistemas de *backup*, garantizando la restauración en caso de un ataque de *ransomware* o pérdida de información. Cada servidor principal de cada entidad médica tendrá un sistema de *backup*, que puede ser externo al sistema o bien puede ser un servicio implementado por la propia Autoridad Certificadora, que tendrá un cifrado propio que garantice la privacidad. La Autoridad Certificadora, que también cuenta con base de datos; contará, asimismo, con una copia de respaldo propia. Se debe proporcionar una buena política de copias de seguridad (incrementalidad, periodicidad, etc.) estableciendo un periodo razonable entre copia y copia en función del volumen de datos.

Se deben establecer mecanismos de seguridad complementarios al sistema, como la implantación de mecanismos de seguridad física (protección del acceso físico, ante desastres naturales o alteraciones del entorno). Por ejemplo, la ubicación física de los servidores del sistema debe garantizar el menor daño en caso de catástrofes como inundaciones, humedad, incendios, humo, etc. Una solución viable a esta problemática consiste en contratar servicios de *hosting* a terceros, responsabilizando a una tercera parte. También se ha de establecer mecanismos de seguridad en red, como puede ser el uso de *whitelists* con las que se autorice la comunicación únicamente a máquinas de confianza.

Además, se deben establecer estrategias de seguridad claras, como la definición de políticas de actualización del *hardware/software*, la realización de auditorías *pentesting* periódicas, el establecimiento de condiciones mínimas de complejidad de contraseñas en el sistema, la definición del periodo de validez de los certificados, la elaboración de una guía de uso del sistema, etc.

6.Conclusiones y líneas futuras

La elaboración de este proyecto ha permitido diseñar e implementar un sistema de gestión de datos sensibles sanitarios de forma segura y constituye una aportación al objetivo de garantizar la privacidad de los usuarios en el intercambio de datos.

El proyecto ha concluido con unos resultados razonablemente satisfactorios, pudiendo completar los objetivos marcados desde un inicio, a pesar del limitado tiempo del que se ha dispuesto. Entre los objetivos cumplidos, se han llevado a cabo algunas de las mejoras y líneas futuras propuestas en el trabajo de fin de grado previo.

El sistema diseñado e implementado ofrece protección frente a ataques a la autenticación de usuarios, utilizando funciones de derivación de clave como Argon2id, que permiten evitar algunos de los ataques actuales más dañinos, como los que emplean granjas de GPU para obtener *hashes* o los ataques de canal lateral.

Se ha fortalecido la gestión de las claves implicadas en los procesos de cifrado y compartición de la información, diseñando e implementando protocolos seguros que hacen uso de criptografía simétrica y asimétrica; optando por algoritmos fiables y robustos como AES256 y RSA con tamaño de 2048 bits. En prevención de posibles accesos indebidos, todas las claves del sistema son almacenadas cifradas.

Se ha garantizado la autenticidad y no repudio en los procesos de compartición de información mediante el uso de certificados con formato X.509, estándar de facto.

En cuanto a la comunicación entre los nodos que conforman el sistema, se ha garantizado la confidencialidad, integridad y autenticidad de la información transmitida empleando la última versión del protocolo de comunicaciones seguras TLS, evitando ataques como la interceptación o alteración de los datos y posibles suplantaciones de identidad.

Aun cuando el objetivo prioritario se ha centrado principalmente en aspectos de seguridad, se ha hecho énfasis en la escalabilidad de la arquitectura del sistema desarrollado debido a las características concretas del modelo al que está destinado.

Utilizar el lenguaje Go constituye una garantía y un punto a favor de la eficiencia del sistema, ya que es un lenguaje que cuenta con librerías criptográficas específicas, eficientes y completas.

Entre las mejoras abordadas, se ha diseñado e implementado un protocolo de intercambio de historiales clínicos entre varias entidades médicas, siendo un punto de especial importancia. Una ampliación natural para esta funcionalidad consiste en hacer que el protocolo sea compatible con servidores desarrollados en otras tecnologías, ya que la implementación actual ha sido diseñada exclusivamente para el entorno de pruebas, implementado en Go.

También se ha implementado un sistema de *logs* centralizado que recoge y clasifica tanto accesos como incidencias y mantiene un orden histórico del sistema. Existen algunos aspectos posibles de ampliación, que sin duda serían desarrollados en un producto profesional como el establecimiento de niveles de alerta en función de la incidencia o la implementación de un sistema de alerta en tiempo real para el administrador.

En resumen, el proyecto desarrollado ha permitido implementar un sistema más seguro que muchos sistemas existentes y que posibilita compartir y almacenar información sensible sanitaria con las máximas garantías.

7.Referencias

[1] «Xataka,» 23 9 2018. [En línea]. Available: https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda.

[2] «DocuSign,» 30 10 2020. [En línea]. Available: https://www.docusign.mx/blog/pki.

[3] «Robo Datos Sanitarios Noruega,» 2019. [En línea]. Available: https://www.incibe-cert.es/alerta-temprana/bitacora-ciberseguridad/helsecert-alerta-robo-datos-sanitarios-mas-3-millones.

[4] «Caso de sustracción de datos sanitarios a mujeres en China,» 2019. [En línea]. Available: https://www.theverge.com/2019/3/11/18260816/china-exposed-database-breedready-women.

[5] «Cesicat,» 2020. [En línea]. Available: https://ciberseguretat.gencat.cat/ca/inici.

[6] «bitcoin.org,» 4 6 2021. [En línea]. Available: https://bitcoin.org/es/.

[7] «ethereum.org,» 5 6 2021. [En línea]. Available: https://ethereum.org/es/.

[8] M. J. L. Lopez, Criptografía y Seguridad en Computadores, 2019.

[9] «firmaelectronica.gob.es,» 5 6 2021. [En línea]. Available: https://firmaelectronica.gob.es/Home/Ciudadanos/DNI-Electronico.html.

[10] «Trello,» 2020. [En línea]. Available: https://trello.com/home.

[11] «Github,» 2020. [En línea]. Available: https://github.com/.

[12] «Google BigQuery,» 2020. [En línea]. Available: https://cloud.google.com/bigquery/?utm_source=google&utm_medium=cpc&utm_campaign=emea-gb-all-en-dr-bkws-all-solutions-trial-e-gcp-1008073&utm_content=text-ad-none-any-DEV_c-CRE_335630920203-

ADGP_Hybrid+%7C+AW+SEM+%7C+BKWS+~+EXA_1:1_GB_EN_Data+Warehousing_.

[13] «Amazon Aurora,» 2020. [En línea]. Available: https://aws.amazon.com/es/rds/aurora/.

[14] «MongoDB,» 2020. [En línea]. Available: https://www.mongodb.com/es.

[15] «Visual Code,» 2020. [En línea]. Available: https://code.visualstudio.com/.

[16] «GitLab,» 2020. [En línea]. Available: https://about.gitlab.com/.

[17] «Go, librería Mysql-Driver,» golang.org, 2020. [En línea]. Available: https://github.com/go-sql-driver/mysql. [Último acceso: 2020].

[18] T. Dierks y E. Rescorla, «The Transport Layer Security (TLS) Protocol,» 2008. [En línea]. Available: https://tools.ietf.org/html/rfc5246.

[19] J. D. a. V. Rijmen, «AES Proposal: Rijndael,» 1999. [En línea]. Available: https://csrc.nist.gov/csrc/media/projects/cryptographic-standards-and-guidelines/documents/aes-development/rijndael-ammended.pdf#page=1.

[20] A. S. a. L. A. R.L. Rivest, «A Method for Obtaining Digital, RSA,» 2020. [En línea]. Available: http://people.csail.mit.edu/rivest/Rsapaper.pdf.

[21] «securityartwork.es,» 2014. [En línea]. Available: https://www.securityartwork.es/2014/04/07/fundamentos-sobre-certificados-digitales-el-estandar-x-509-y-estructura-de-certificados/.

[22] A. J. Menezes, P. C. van Oorschot y S. A. Vanstone, « Handbook of Applied Cryptography,» 1996. [En línea]. Available: https://archive.org/details/handbookofapplie0000mene.

[23] «Salted Password Hashing - Doing it Right,» 2019. [En línea]. Available: https://crackstation.net/hashing-security.htm#salt.

[24] D. D. a. D. K. Alex Biryukov, «Argon2: the memory-hard function for password hashing and other,» 2017. [En línea]. Available: https://www.cryptolux.org/images/0/0d/Argon2.pdf.

[25] «Password hashing,» [En línea]. Available: https://password-hashing.net/.

[26] «Go, librería crypto/x509,» 2021. [En línea]. Available: https://golang.org/pkg/crypto/x509/.

[27] «Go, librería crypto/rsa,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/crypto/rsa/. [Último acceso: 2020].

[28] «What is Base64?,» 2020. [En línea]. Available: https://base64.guru/learn/what-is-base64.

[29] «Go, librería crypto/argon2,» golang.org, 2020. [En línea]. Available: https://godoc.org/golang.org/x/crypto/argon2. [Último acceso: 2020].

[30] «Go, librería crypto/aes,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/crypto/aes/. [Último acceso: 2020].

[31] «Go, librería crypto/cipher,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/crypto/cipher. [Último acceso: 2020].

[32] «Go, librería crypto/sha256,» 2021. [En línea]. Available: https://golang.org/pkg/crypto/sha256/.

[33] «Go, librería crypto/sha512,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/crypto/sha512/. [Último acceso: 2020].

[34] «Go, librería encoding/base64,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/encoding/base64/. [Último acceso: 2020].

[35] E. B. W. B. a. L. C. Meltem Sönmez Turan, «Recommendation for Password-Based Key,» 2010. [En línea]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf.

[36] «Go, librería crypto/tls,» golang.org, 2020. [En línea]. Available: https://golang.org/pkg/crypto/tls/. [Último acceso: 2020].

Apéndice

Figura 12.- Diagrama Entidad-Relación de la base de datos de la Autoridad Certificadora

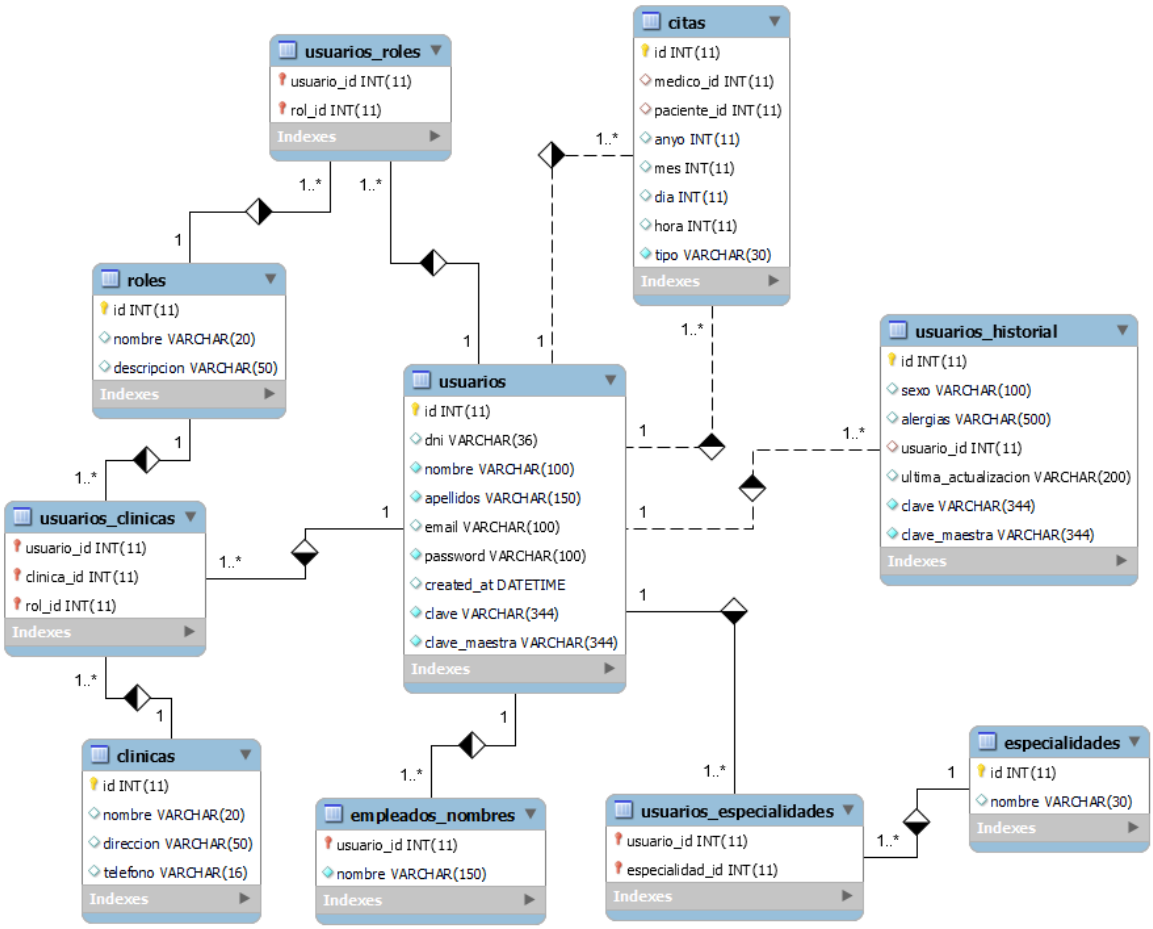

Figura 13.- Diagrama Entidad-Relación de los atributos que definen a un usuario

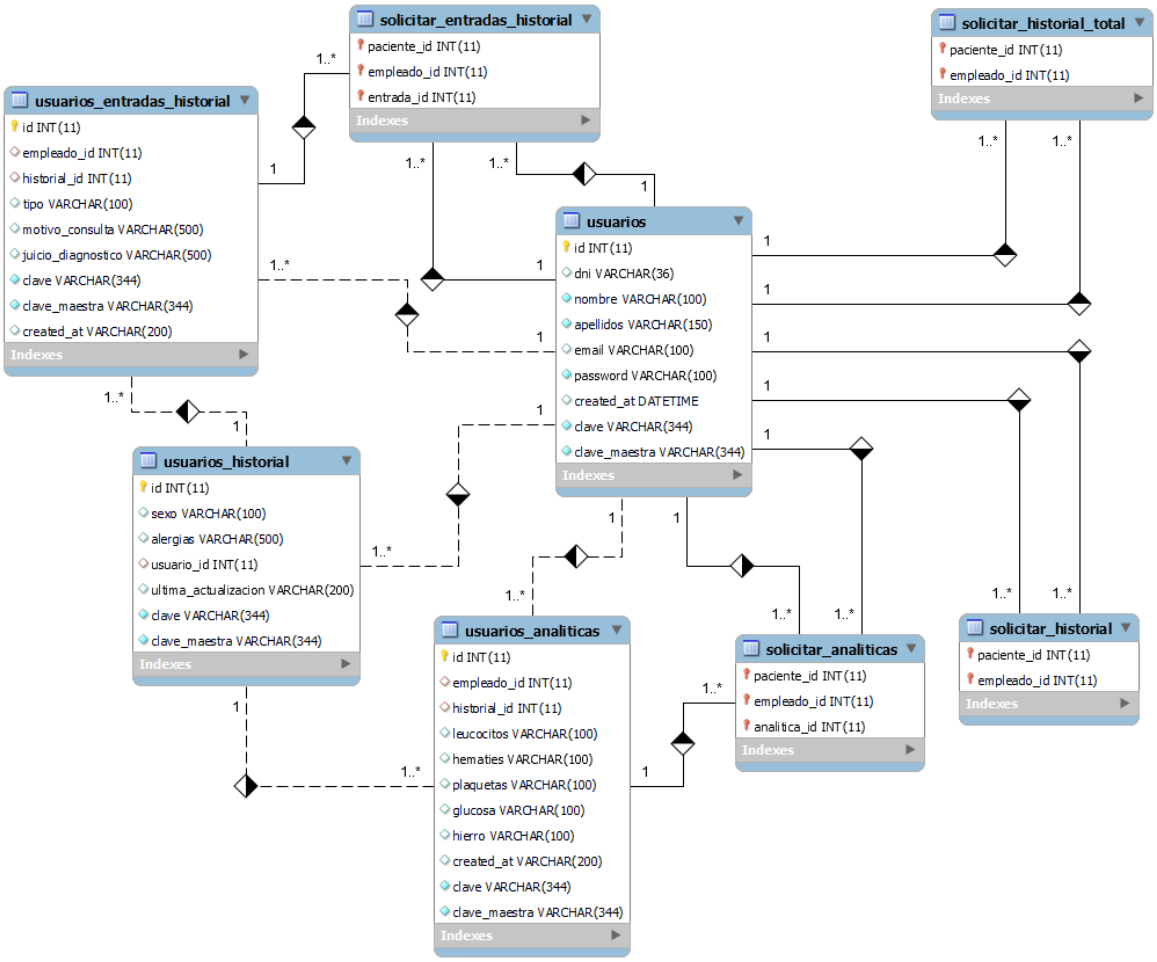

Figura 14.- Diagrama Entidad-Relación de las solicitudes de permisos en el sistema

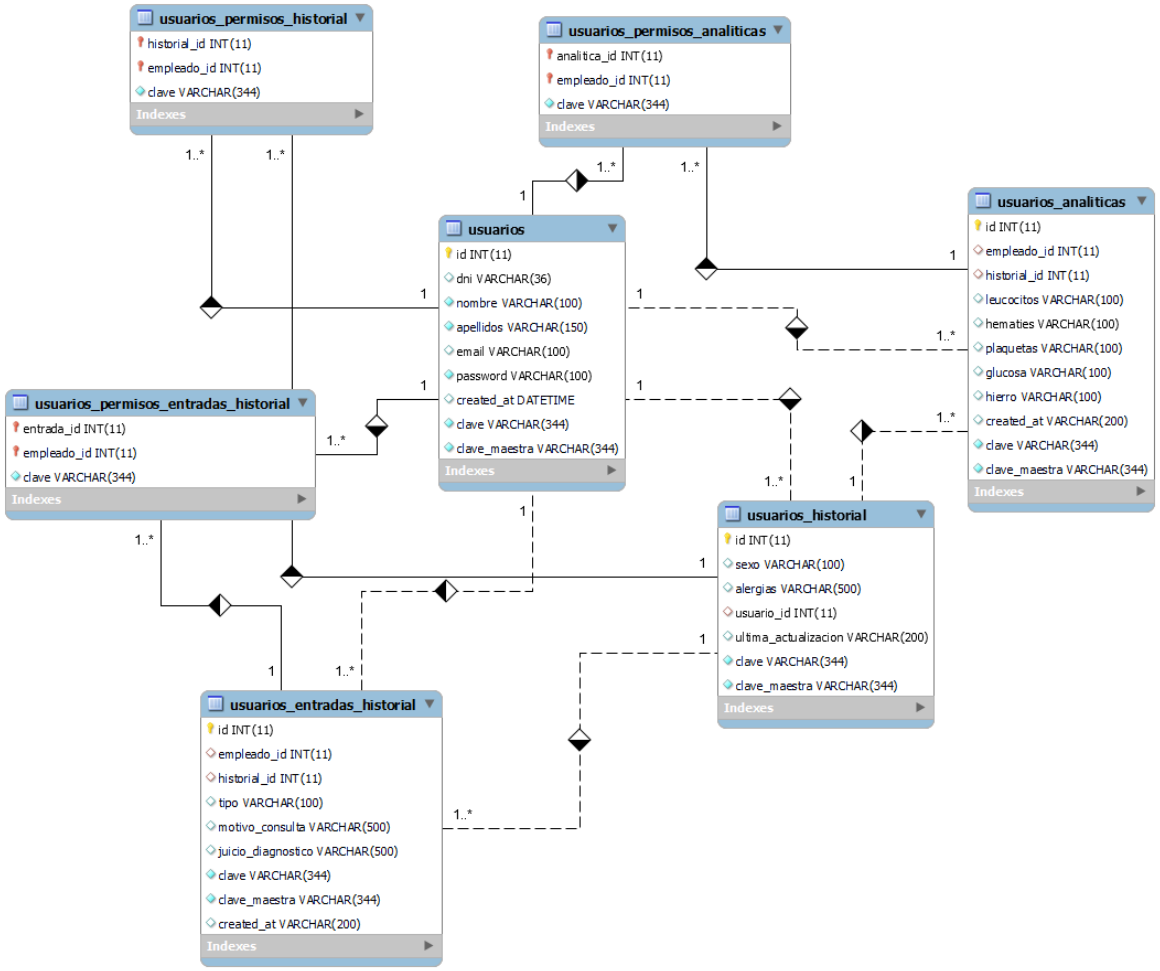

Figura 15.- Diagrama Entidad-Relación del sistema de permisos de la historia clínica

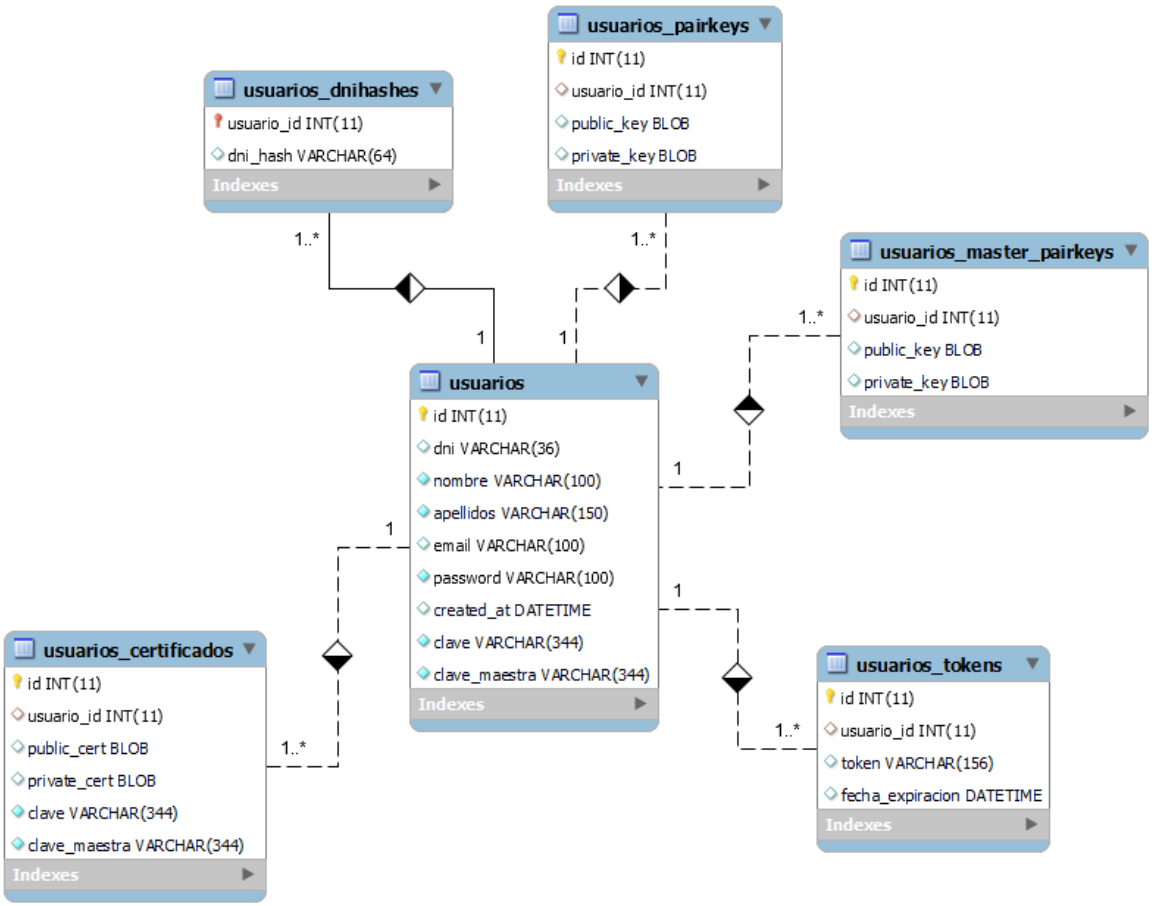

Figura 16.- Diagrama Entidad-Relación de la gestión de claves y certificados del usuario

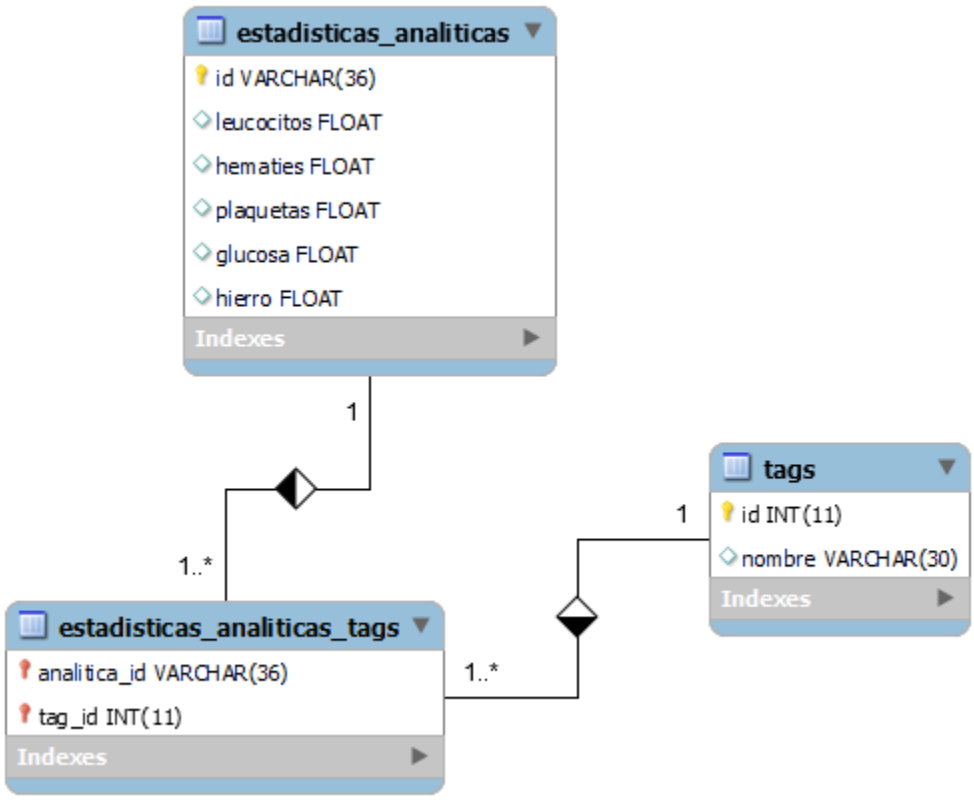

Figura 17.- Diagrama Entidad-Relación del almacenamiento de analíticas anónimas

Glosario

***Man in the middle**:* En el ámbito de la criptografía, ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad. El atacante ha de observar/interceptar los mensajes entre dos nodos y tratar de que ninguna de las víctimas sea consciente de que la comunicación ha sido comprometida.

NIST: *National Institute of Standards and Technology* (en español, Instituto Nacional de Estándares y Tecnología) es una agencia de la Administración de Tecnología del Departamento de Comercio de los Estados Unidos, cuya misión es fomentar la innovación y la competencia industrial en dicho país mediante avances en normas, tecnología o metrología.

PKI: *Public Key Infraestructure* (en español, infraestructura de clave pública) es una combinación de hardware, software y procedimientos de seguridad que garantizan la ejecución de operaciones criptográficas con garantías, como la firma digital, el no repudio o el cifrado.

POST: es un método de la arquitectura REST utilizado para solicitar la creación de un nuevo registro en una colección de recursos.

REST: REpresentational State Transfer es una arquitectura que proporciona estándares entre sistemas informáticos en la web, facilitando la comunicación entre varios sistemas. Un sistema RESTful es un sistema compatible con REST.

RFC: son una serie de publicaciones del IETF (Grupo de Trabajo de Ingeniería de Internet en español) en las que se detallan diversos aspectos del funcionamiento de Internet y otras redes de computadoras, como procedimientos, protocolos, etc.

Sniffing: Técnica utilizada para escuchar todo lo que ocurre en una red de computadores, normalmente a través de herramientas denominadas *sniffers*.

SQL: Es un lenguaje utilizado en programación, diseñado para administrar información de bases de datos relacionales.

SSL: *Secure Sockets Layer* (en español, capa de conexión segura), es el nombre que recibe un protocolo empleado para realizar conexiones seguras en red mediante el uso de certificados y criptografía asimétrica.